Swimming with Shark

Session 407

These are confidential sessions—please refrain from filming or taking pictures.

Apple Applications

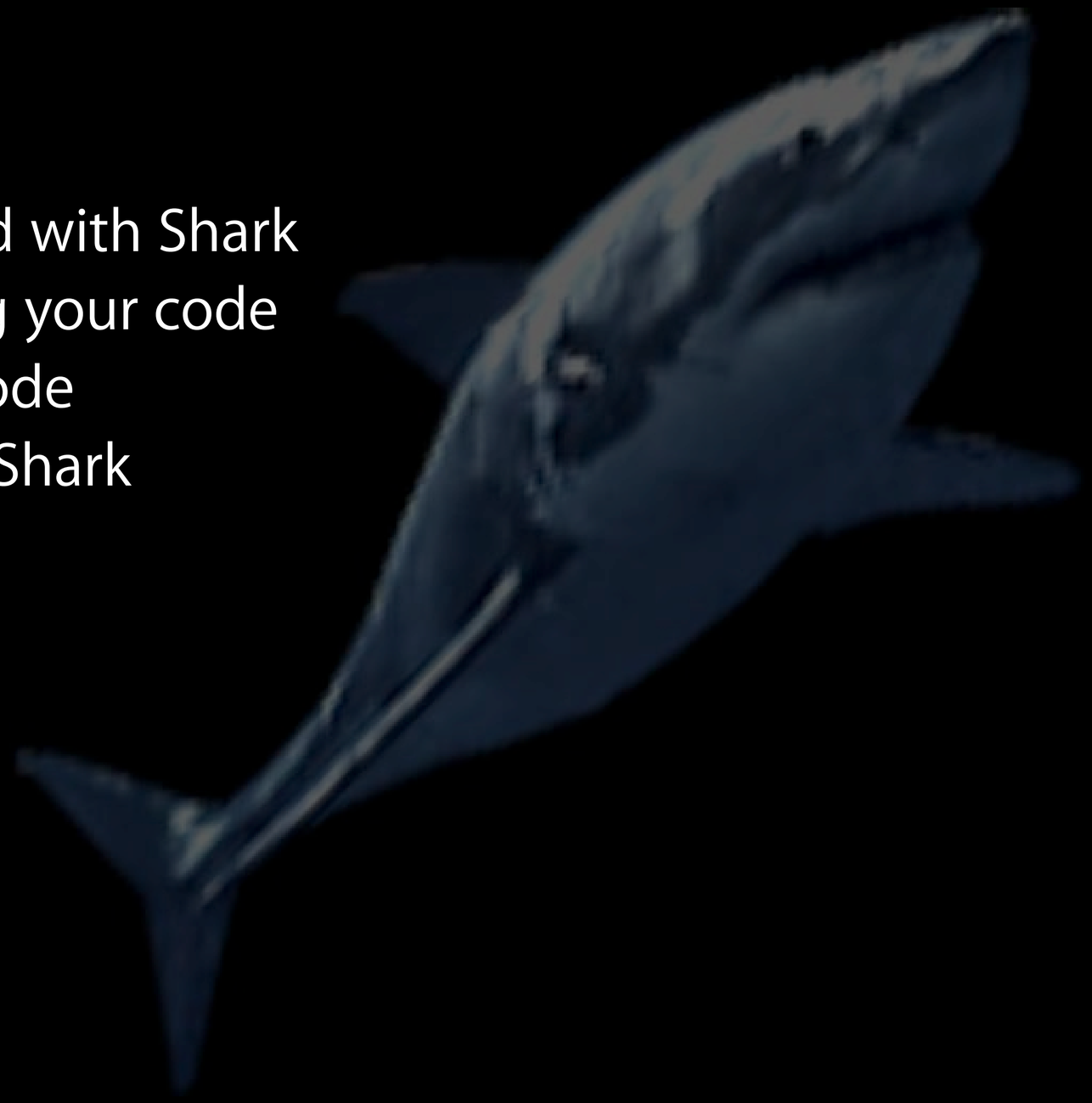
Swimming with Shark

Nathan Slingerland
Sanjay Patel
Architecture and Performance Group

https://developer.apple.com/wwdc2005/

Overview

- What is Shark?
- Getting started with Shark
- Understanding your code
- Tuning your code
- A new way to Shark



What Is Shark?

https://developer.apple.com/wwdc2005/

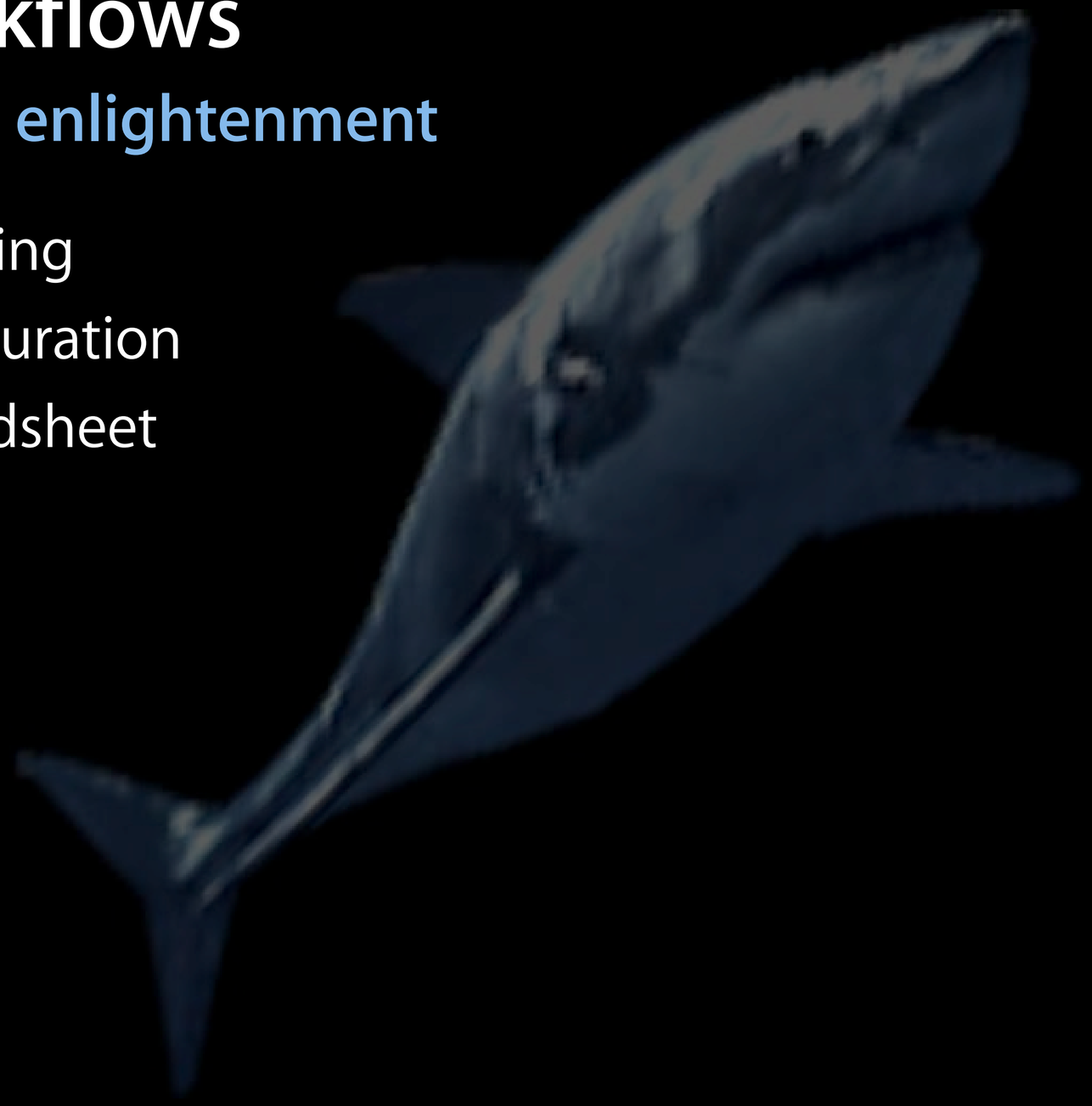
Shark

- Simple and fast profiling tool for Mac OS X
- Helps you find and fix performance problems
- Works with the language and compiler of your choice
  - C, C++, ObjC, Fortran, Java
  - GCC, CodeWarrior, XLC/XLF, Absoft Fortran
  - Mach-O, CFM
- GUI or command-line
- CHUD 4.2 Preview available today
  - http://developer.apple.com/tools/download/
- Free!

Shark Workflows

Many paths to enlightenment

- Network profiling
- Custom configuration
- Counter spreadsheet
- Malloc tracing



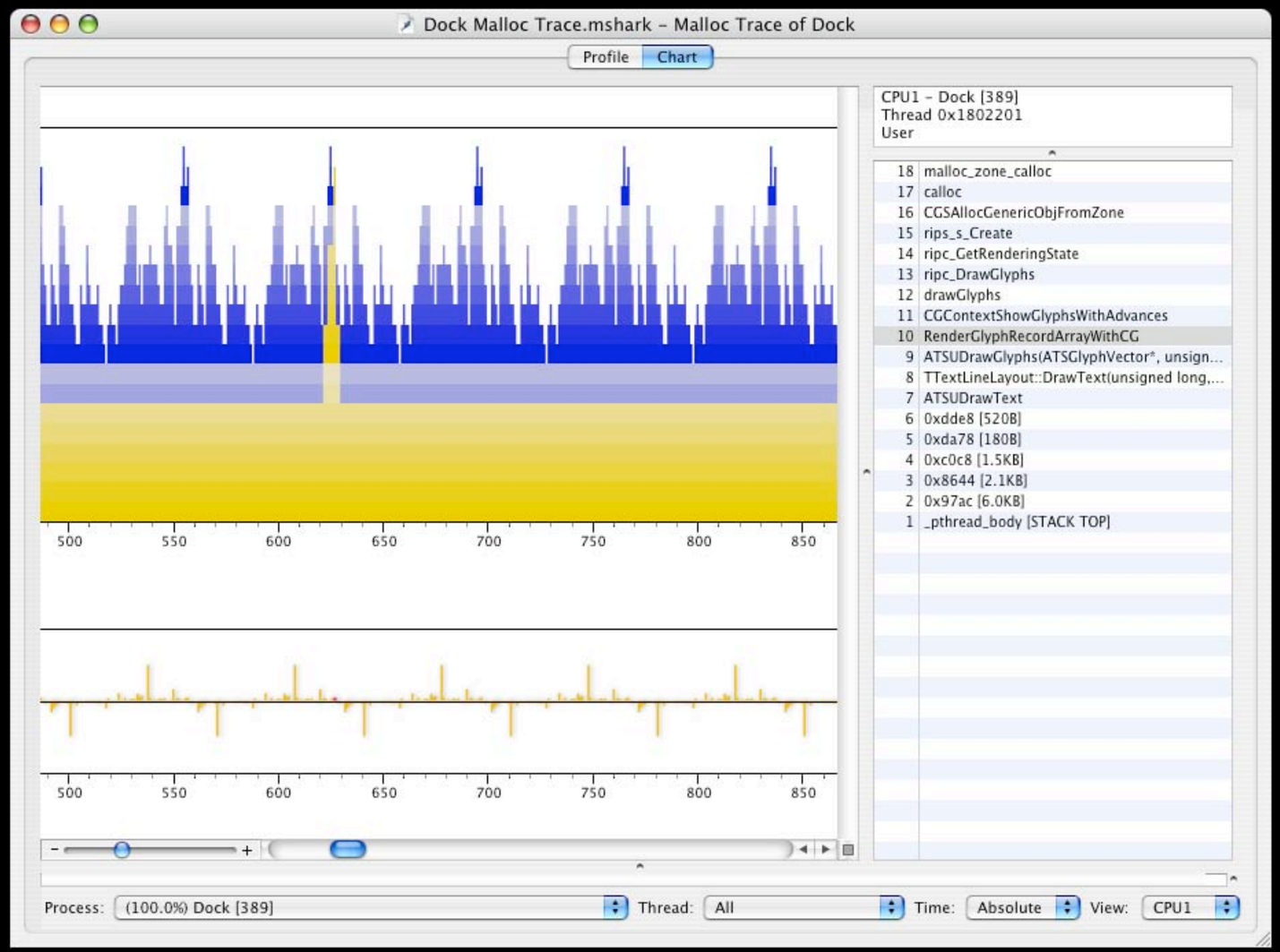
Time Profile

A brief history of your computer

- The most common workflow
  - Shows you where time is being spent
  - Optimize performance-critical code
- Statistical sampling
  - 1,000 data points every second (default)
  - Each sample contains backtrace, process, thread, etc.
- Very low overhead
- Captures everything—drivers, kernel, applications

Malloc Tracing

Thin is in

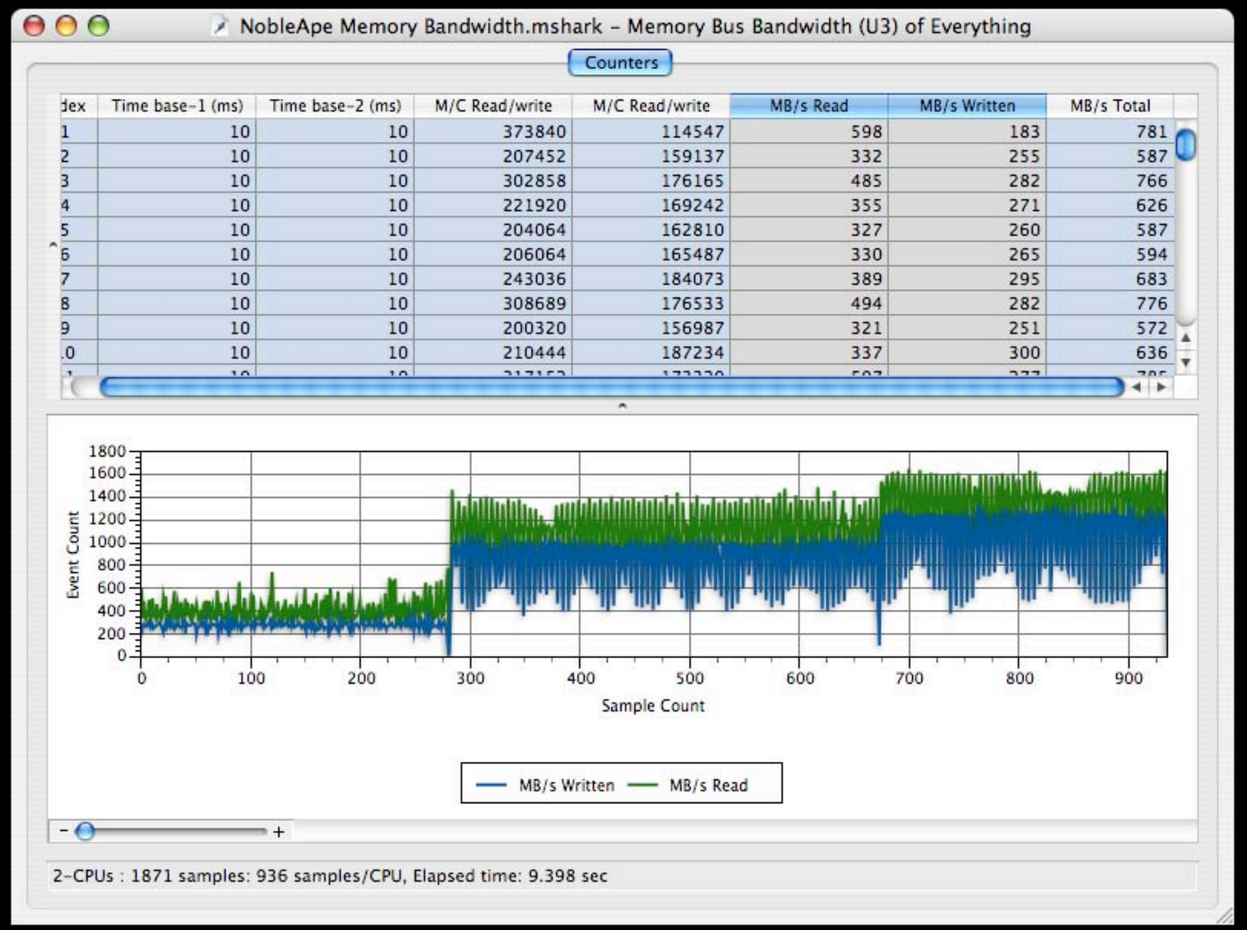


- Monitor allocations (`malloc`, `new`, etc.)
- Understand application memory usage
- Visualize program behavior

Counter Spreadsheet

Add it all up

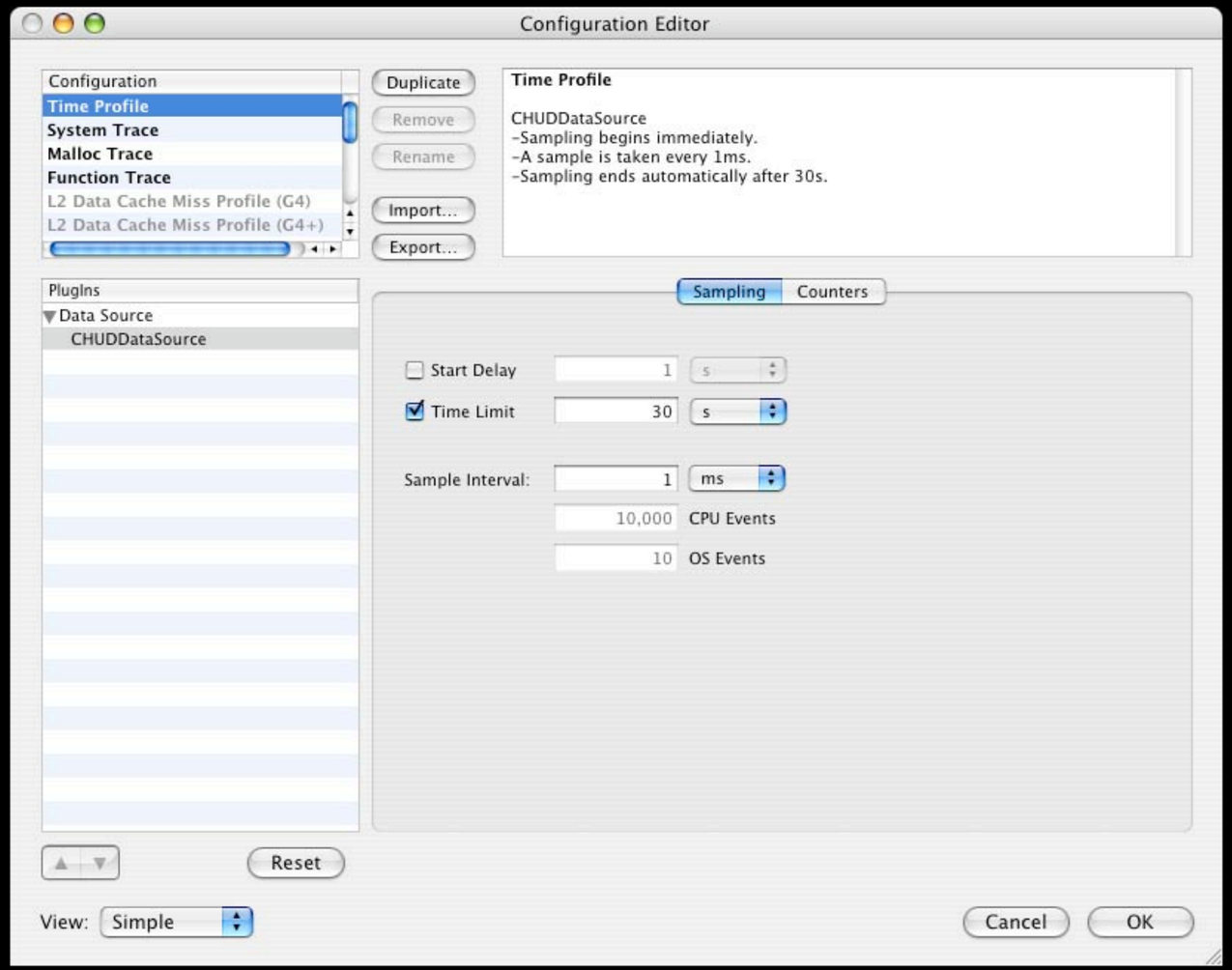
- Quantify performance events (memory bandwidth, CPI)
- Design your own metrics



Configuration Editor

Do it yourself

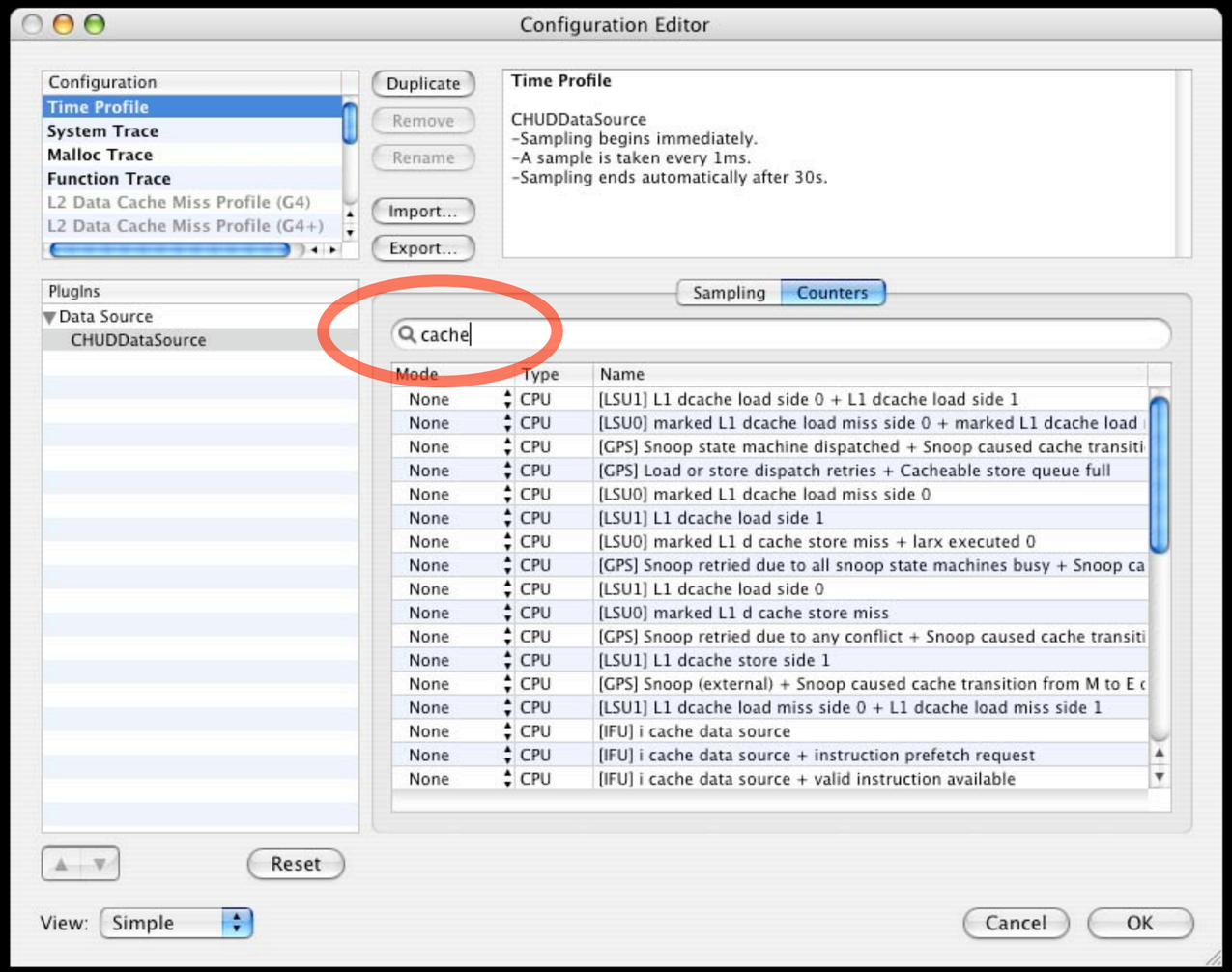
- Create your own configs



Configuration Editor

Do it yourself

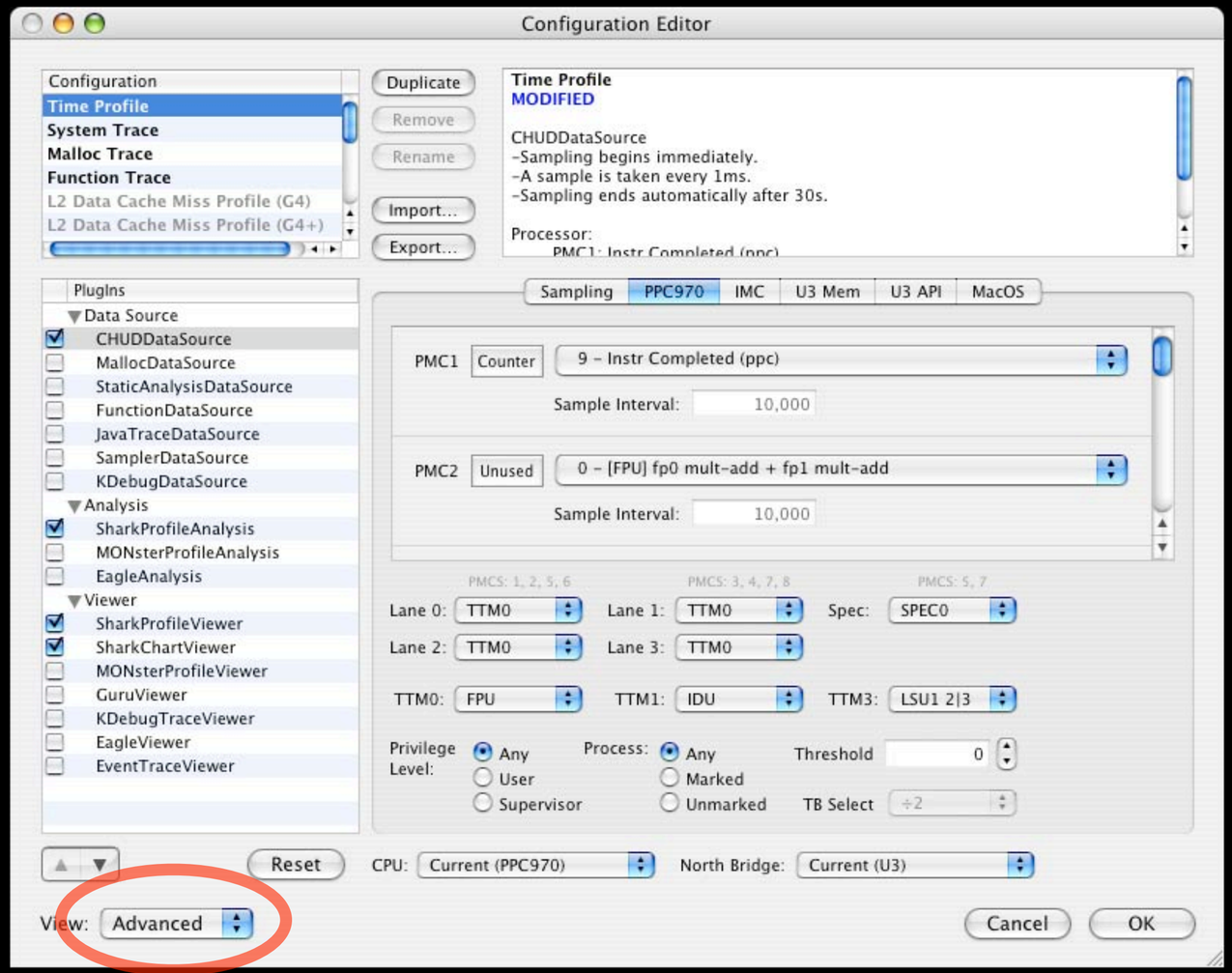
- Create your own configs
- Search event lists by name



Configuration Editor

Do it yourself

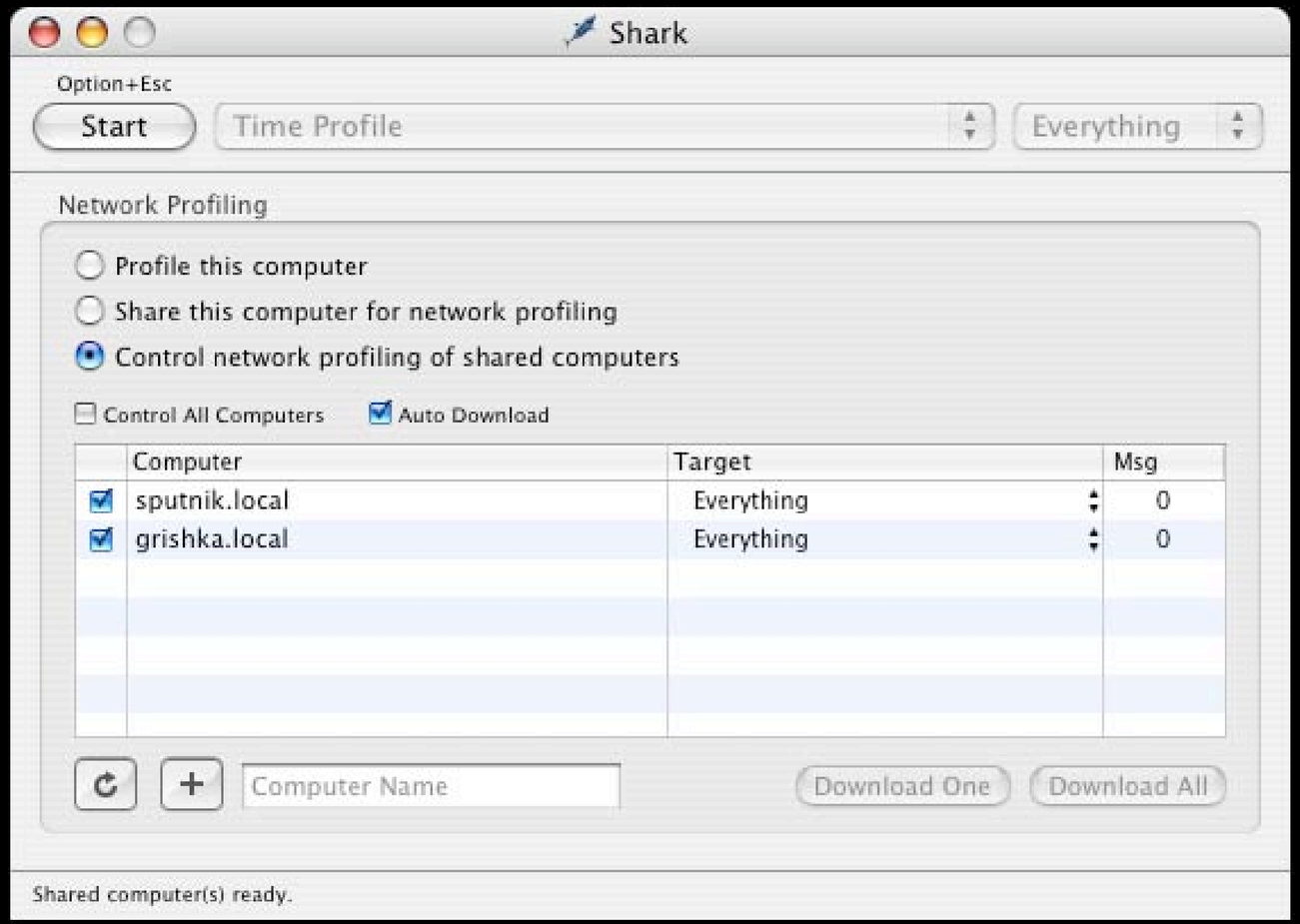
- Create your own configs
- Search event lists by name
- Control every aspect of profiling!



Network Profiling

The world at your fingertips

- Use Bonjour or TCP/IP to track remote systems
- Simultaneously profile multiple machines



Getting Started With Shark

https://developer.apple.com/wwdc2005/

Performance Analysis 101

Step 1:

Establish a Baseline

Step 2:

Profile Optimized Deployment Code With Debug Symbols

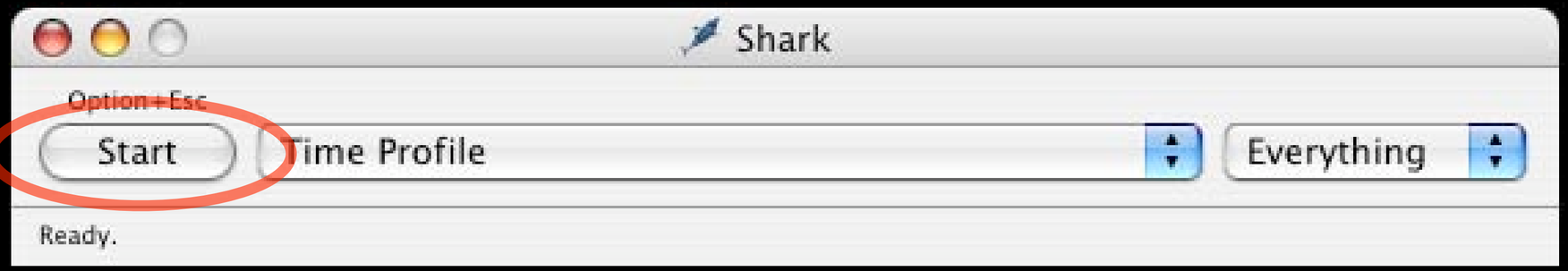
Step 3:

Shark It!

Getting Started With Shark

Just one click

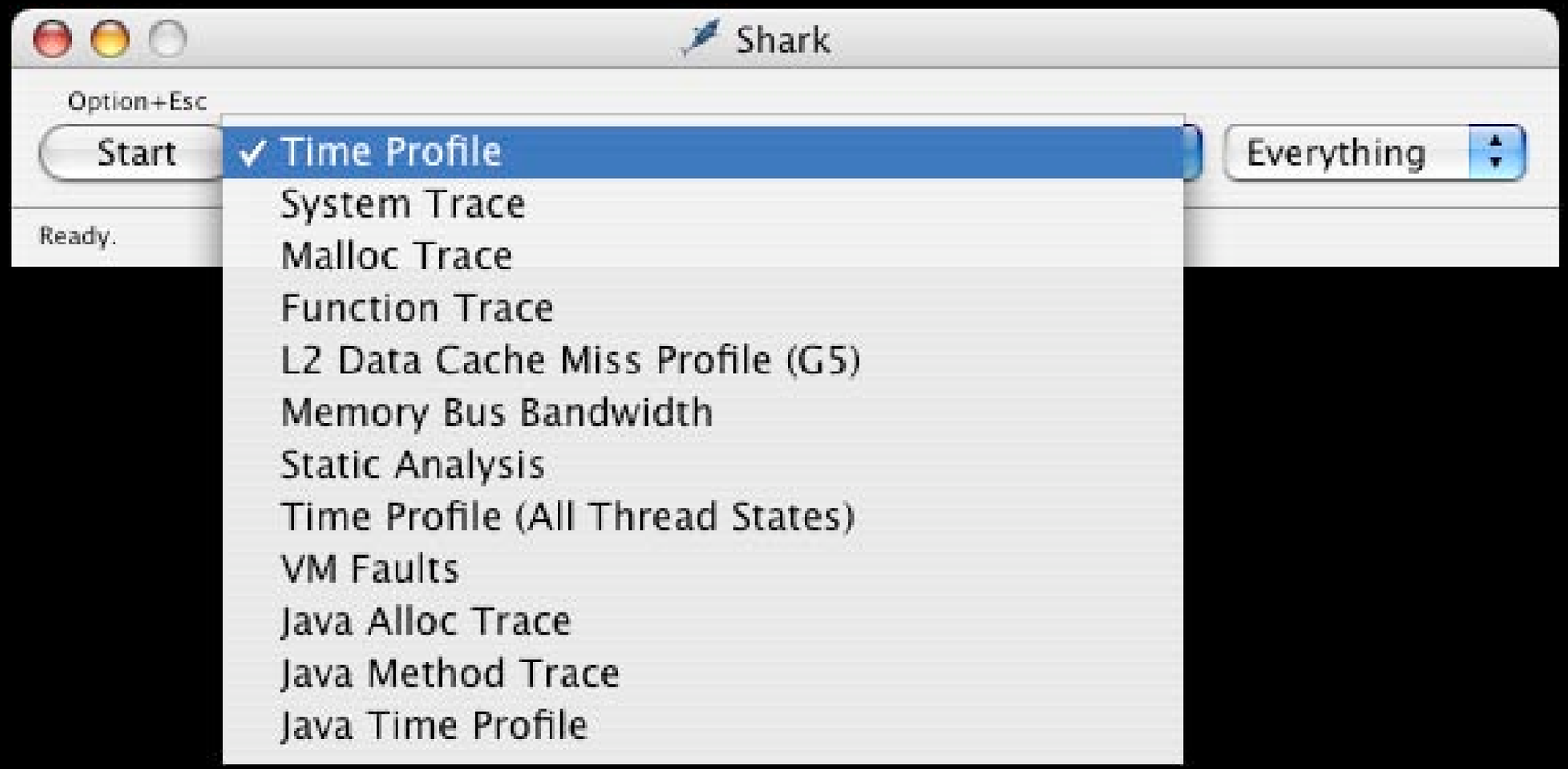
- Use Start button or hot key to begin profiling
- You're on your way…



Getting Started With Shark

Choose your weapon

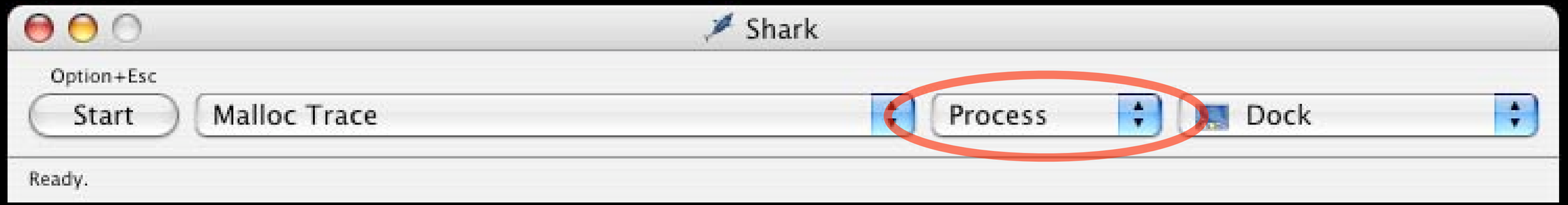
- Sampling configurations



Getting Started With Shark

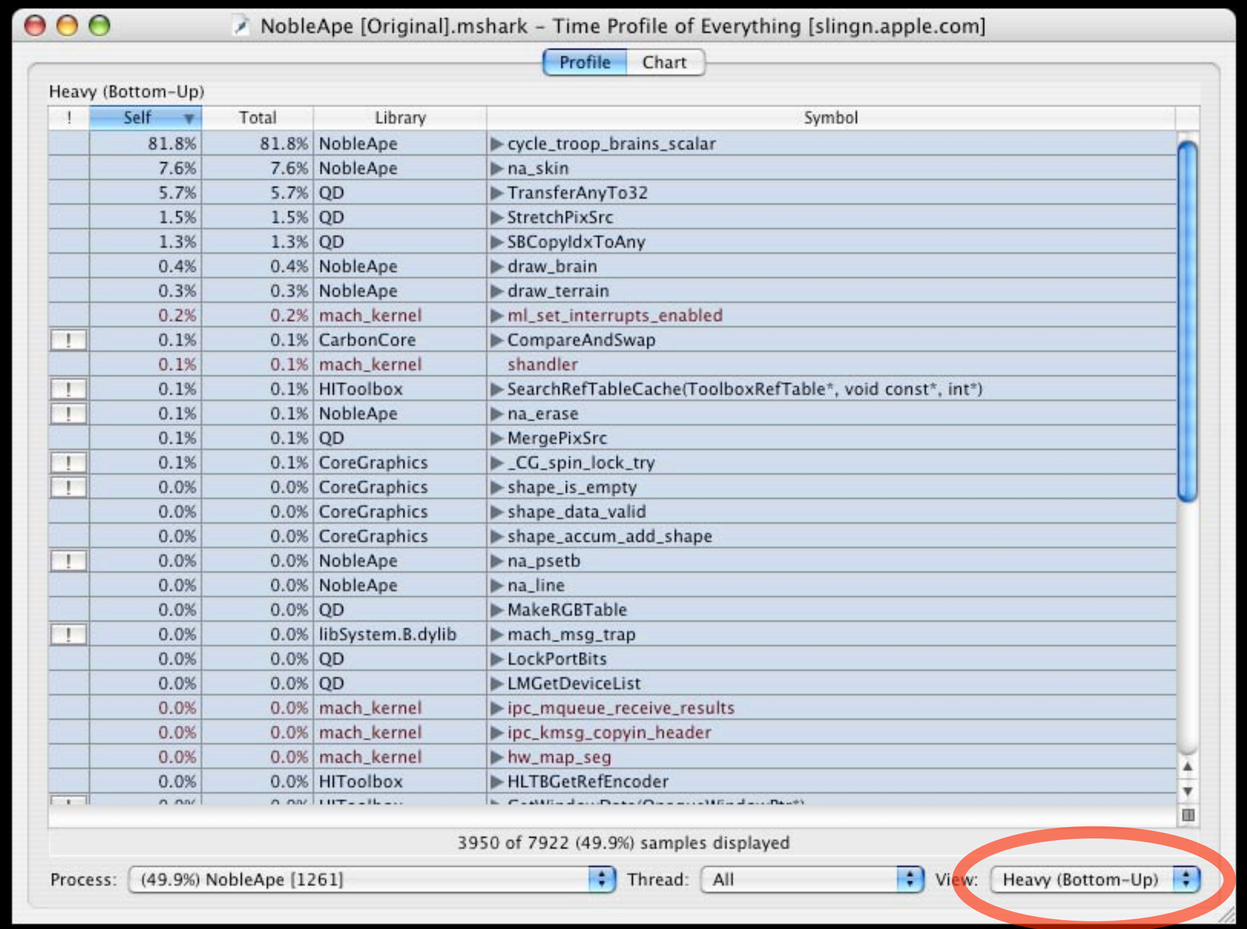
Take aim

- Target everything, process, or file



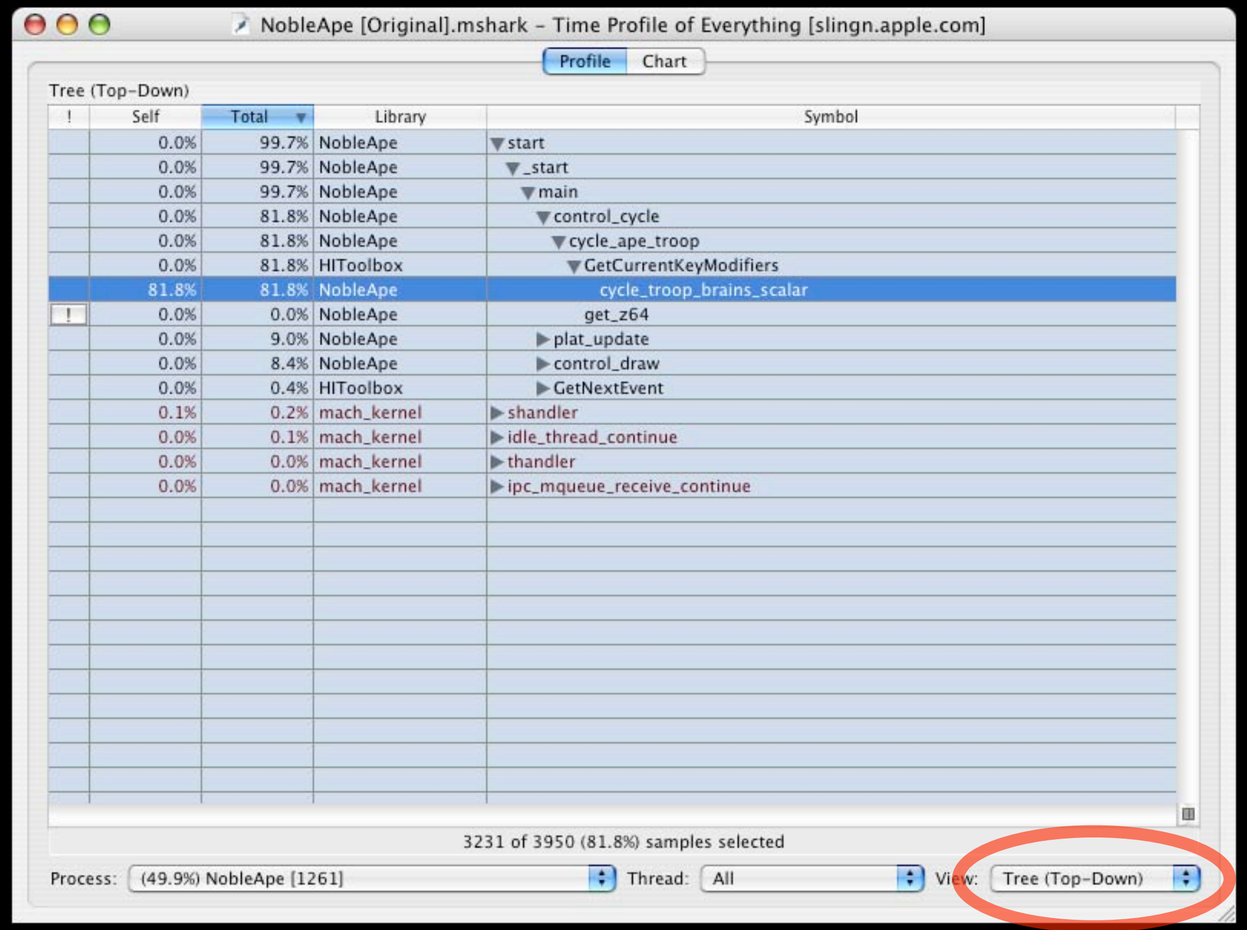
Profile Browser—Heavy View

The bottom line



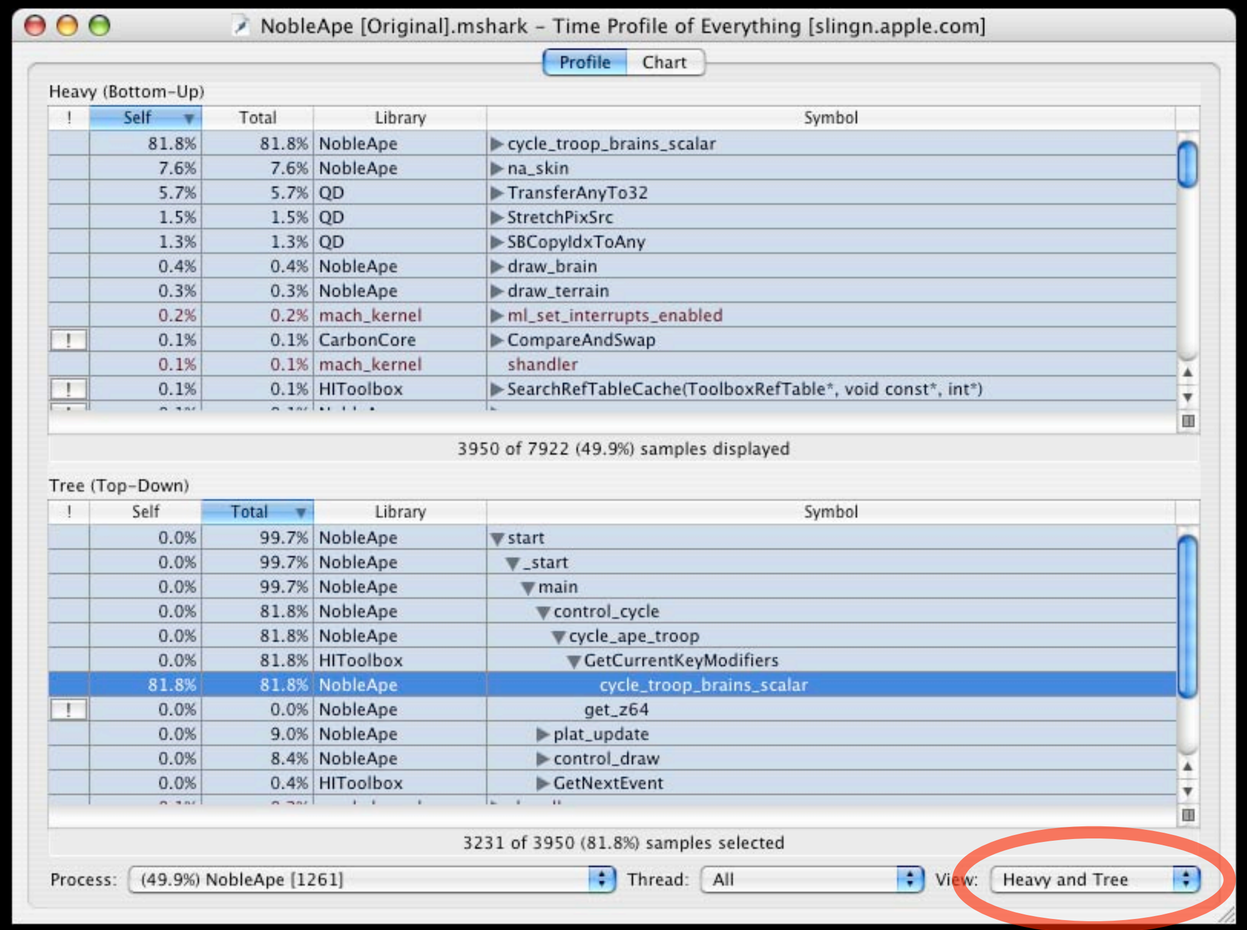
Profile Browser—Tree View

Getting there is half the fun



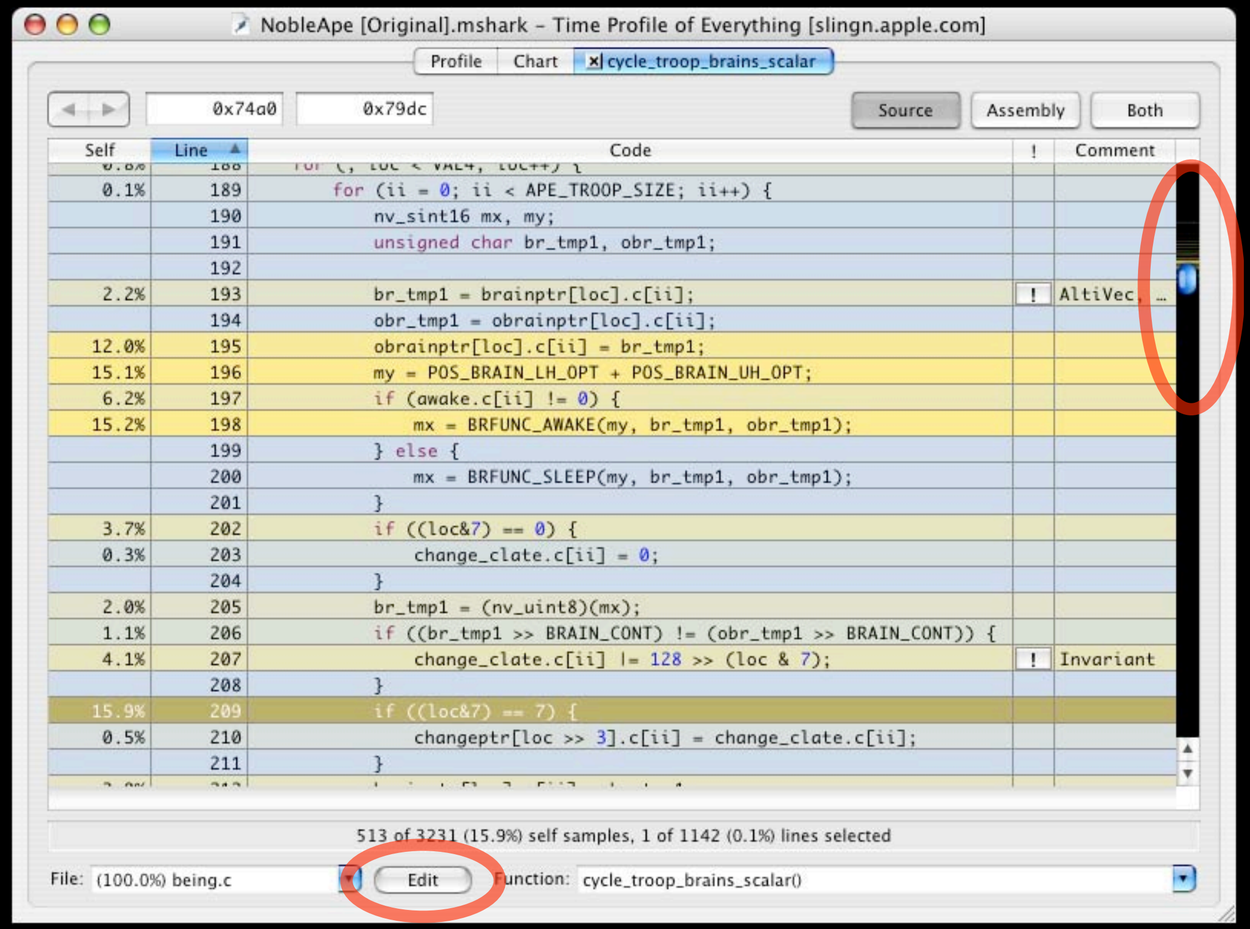
Profile Browser—Heavy and Tree

Where am I and how did I get here?



Code Browser

Here comes the sun

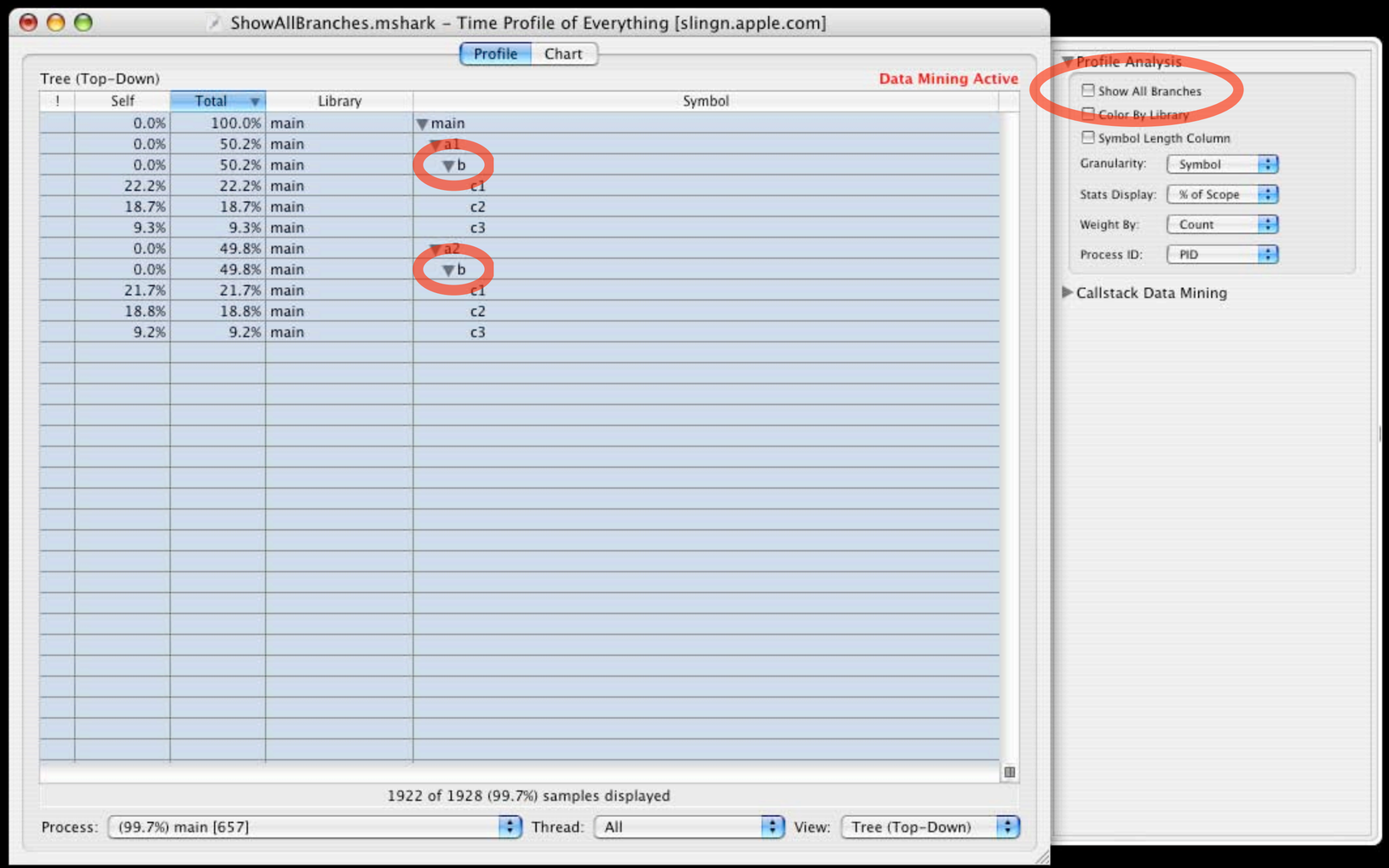


Understanding Your Code

https://developer.apple.com/wwdc2005/

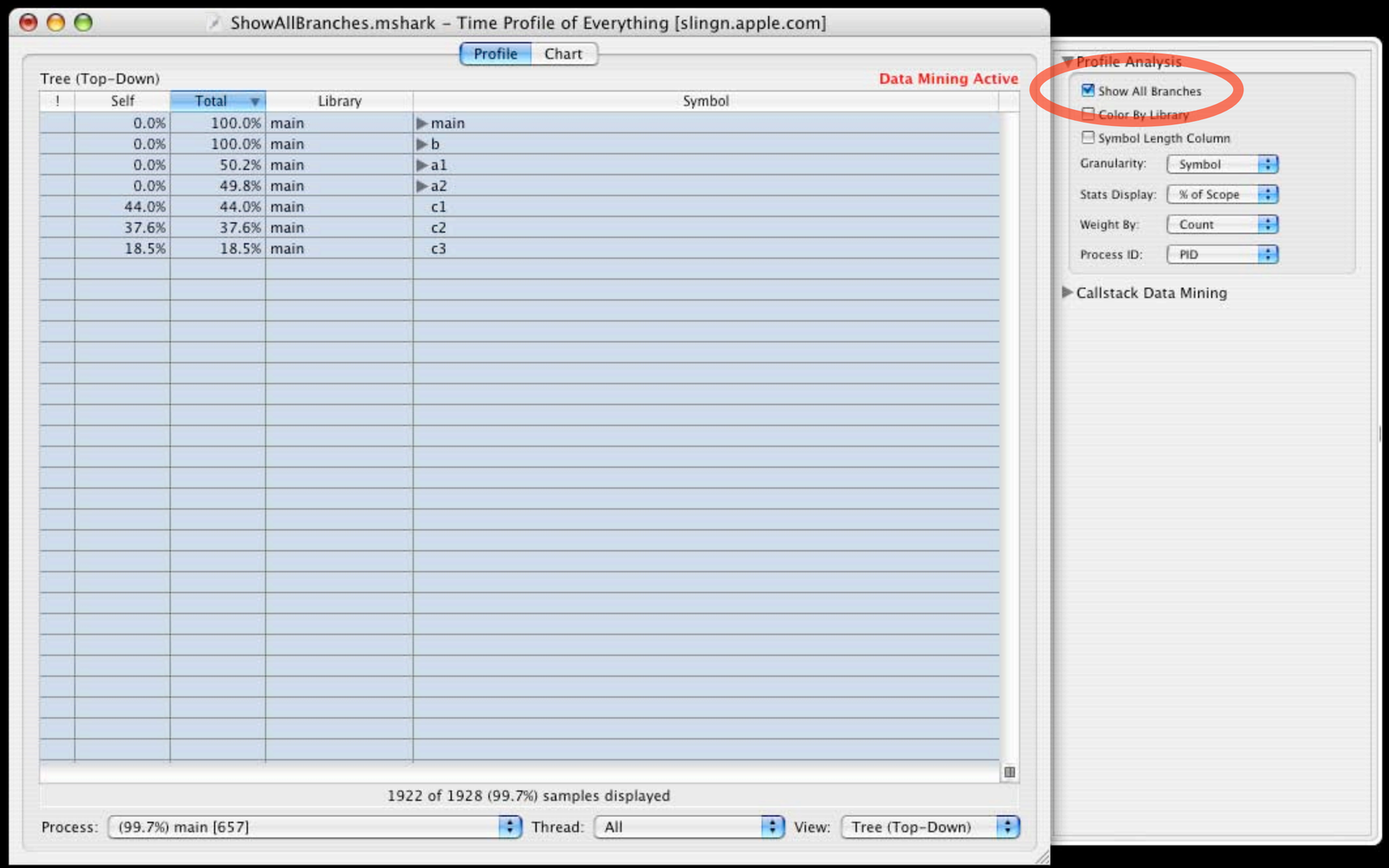
Profile Analysis

Welcome to the jungle



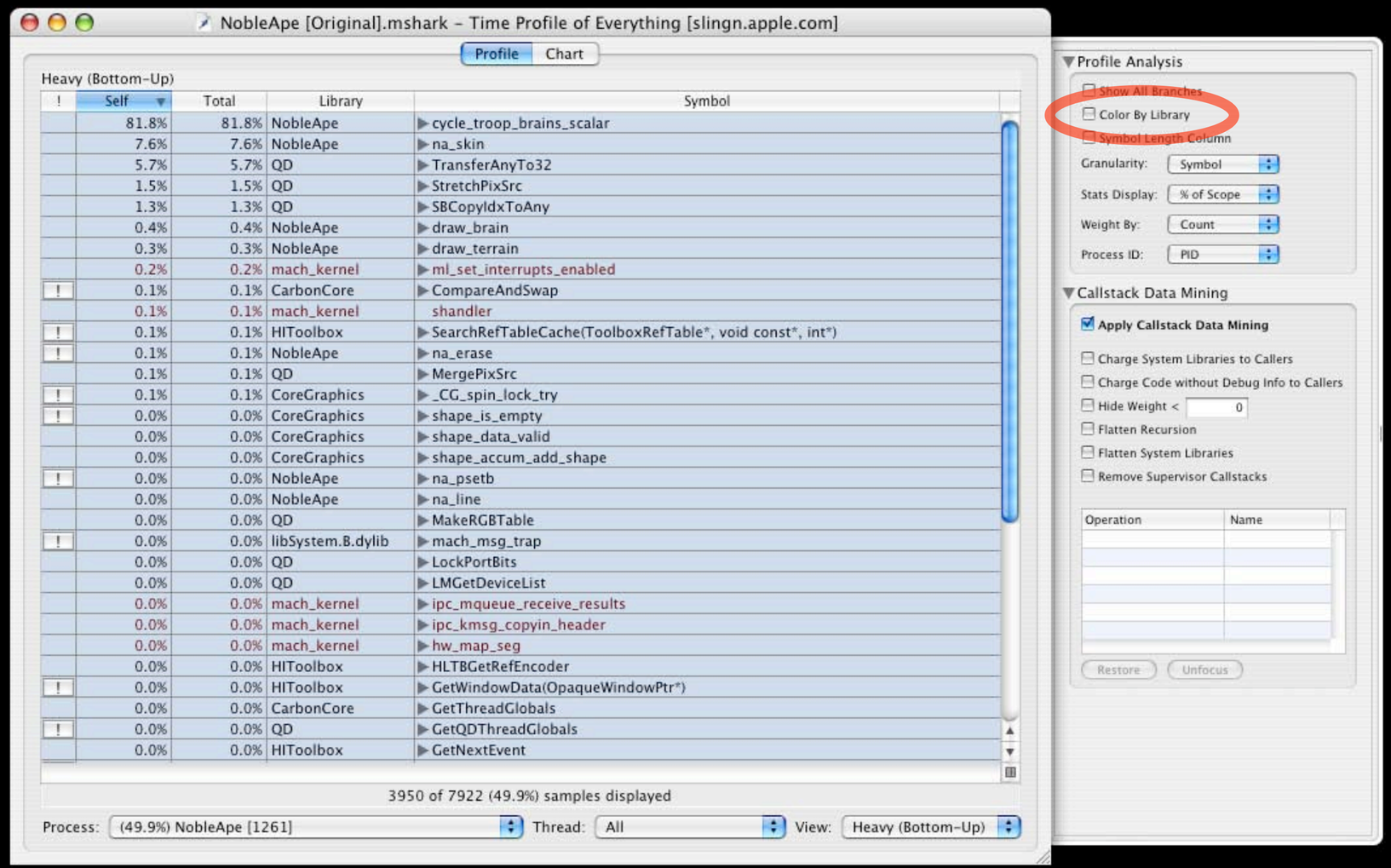
Profile Analysis

Welcome to the jungle



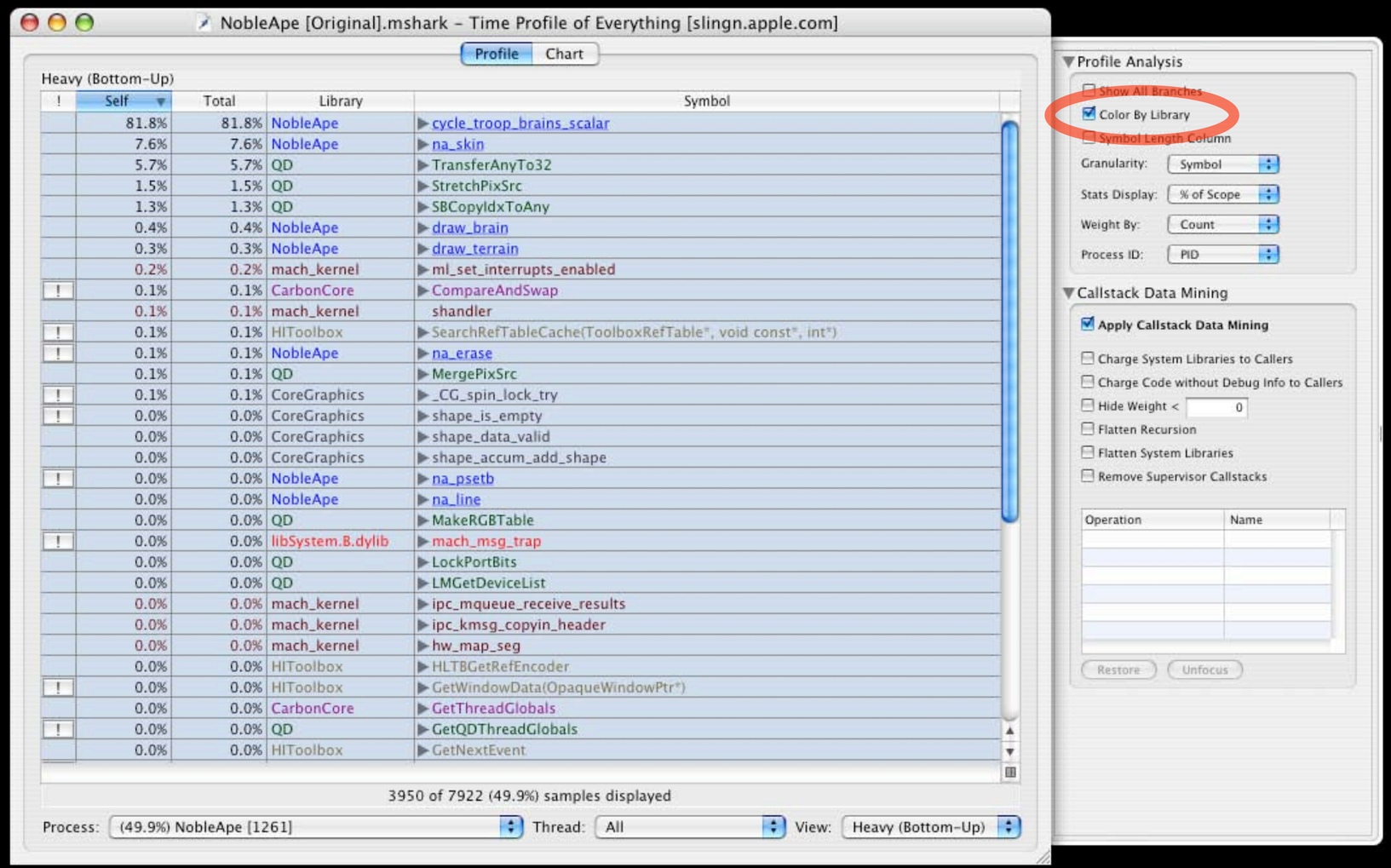
Profile Analysis

A rainbow of possibilities



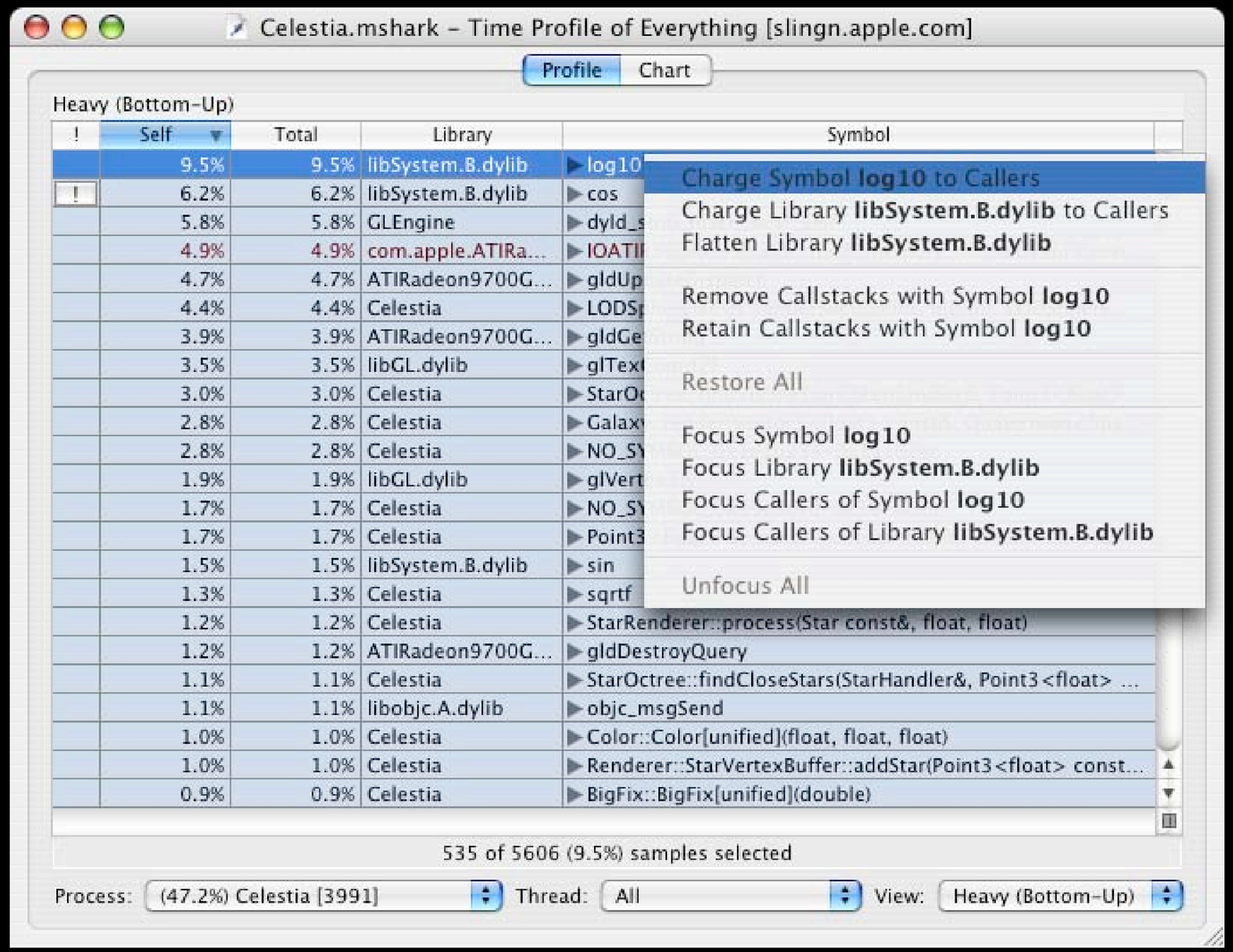
Profile Analysis

A rainbow of possibilities



Data Mining

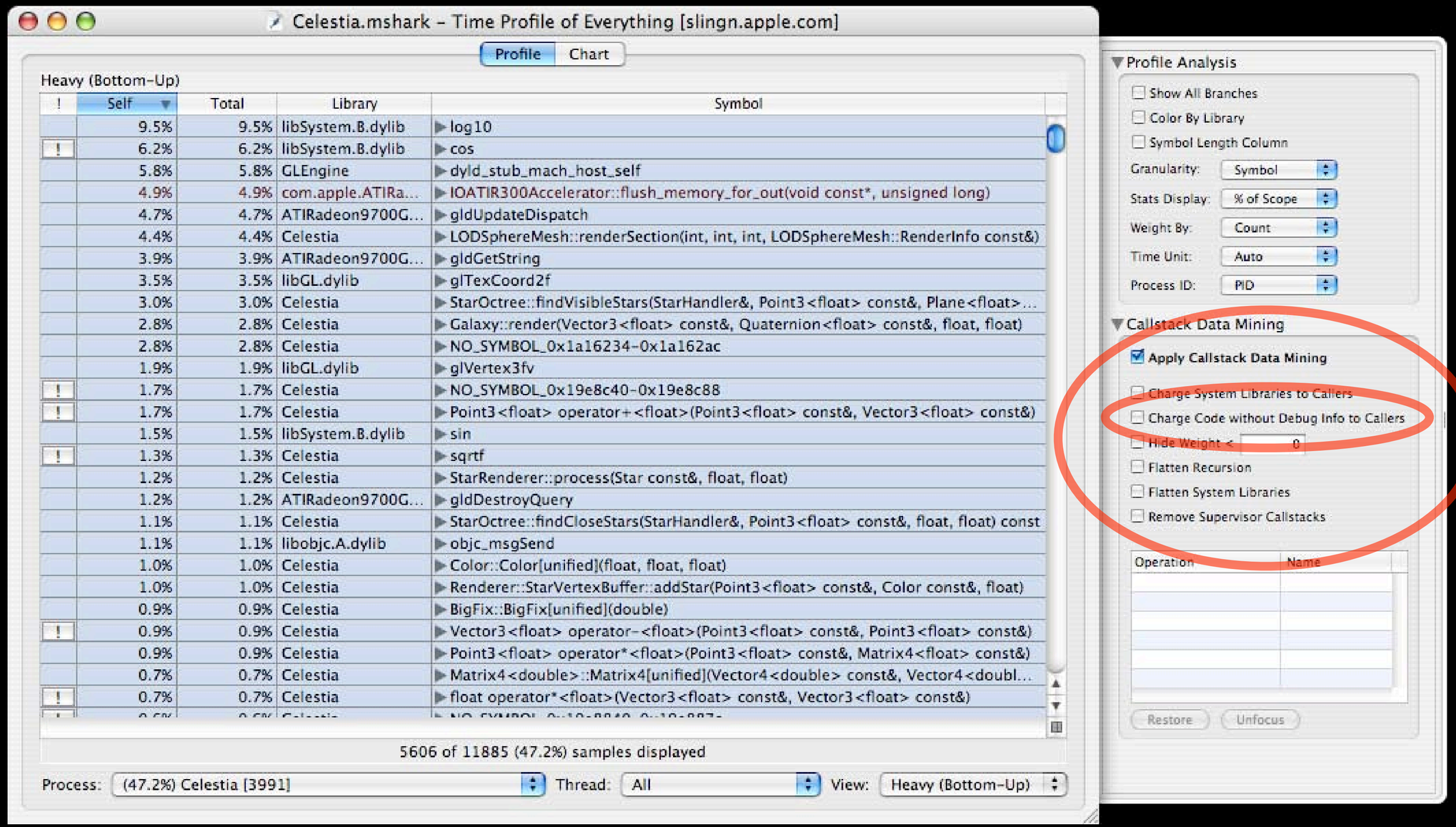
Session surgery



- Charge Symbol/Library
  Hide `foo`; assign samples to caller
- Focus Symbol
  Show paths starting from `foo`
- Flatten Library
  Show only entry points to `libFoo`

Data Mining

Out of sight, out of mind



Data Mining

Out of sight, out of mind

Celestia.mshark – Time Profile of Everything [slingn.apple.com]

Profile | Chart

Heavy (Bottom-Up) — Data Mining Active

!	Self	Total	Library	Symbol
	16.2%	16.2%	Celestia	▶LODSphereMesh::renderSection(int, int, int, LODSphereMesh::RenderInfo const&)
	14.8%	14.8%	Celestia	▶StarOctree::findVisibleStars(StarHandler&, Point3<float> const&, Plane<float>…
	14.1%	14.1%	Celestia	▶Galaxy::render(Vector3<float> const&, Quaternion<float> const&, float, float)
	6.6%	6.6%	Celestia	▶VSOP87Orbit::computePosition(double) const
	4.8%	4.8%	Celestia	▶Renderer::StarVertexBuffer::addStar(Point3<float> const&, Color const&, float)
	4.2%	4.2%	Celestia	▶TextureFont::render(wchar_t) const
	2.2%	2.2%	Celestia	▶Renderer::StarVertexBuffer::finish()
	2.1%	2.1%	Celestia	▶sqrtf
	2.0%	2.0%	Celestia	▶Point3<float> operator+<float>(Point3<float> const&, Vector3<float> const&)
	1.9%	1.9%	Celestia	▶StarRenderer::process(Star const&, float, float)
	1.5%	1.5%	Celestia	▶Renderer::render(Observer const&, Universe const&, float, Selection const&)
	1.3%	1.3%	Celestia	▶StarOctree::findCloseStars(StarHandler&, Point3<float> const&, float, float) const
	1.3%	1.3%	Celestia	▶std::pair<double, double> solve_iteration_fixed<double, SolveKeplerFunc2>(…
	1.3%	1.3%	Celestia	▶Color::Color[unified](float, float, float)
	1.2%	1.2%	Celestia	▶BigFix::BigFix[unified](double)
	1.1%	1.1%	Celestia	▶TextureFont::render(std::basic_string<char, std::char_traits<char>, std::alloca…
!	1.1%	1.1%	Celestia	▶Vector3<float> operator-<float>(Point3<float> const&, Point3<float> const&)
	1.1%	1.1%	Celestia	▶Renderer::renderEllipsoidAtmosphere(Atmosphere const&, Point3<float>, Qua…
	1.0%	1.0%	Celestia	▶Point3<float> operator*<float>(Point3<float> const&, Matrix4<float> const&)
	0.8%	0.8%	Celestia	▶Matrix4<double>::Matrix4[unified](Vector4<double> const&, Vector4<doubl…
	0.8%	0.8%	Celestia	▶Matrix3<double>::yrotation(double)
!	0.8%	0.8%	Celestia	▶float operator*<float>(Vector3<float> const&, Vector3<float> const&)
!	0.7%	0.7%	Celestia	▶Color::Color[unified](Color&, float)
!	0.7%	0.7%	Celestia	▶float clamp<float>(float)
!	0.6%	0.6%	Celestia	▶Vector3<float> operator*<float>(Vector3<float> const&, float)
	0.5%	0.5%	Celestia	▶EllipticalOrbit::positionAtE(double) const
!	0.5%	0.5%	Celestia	▶astro::absToAppMag(float, float)

4609 of 4612 (99.9%) samples displayed

Process: (99.9%) Celestia [3991] Thread: All View: Heavy (Bottom-Up)

▼Profile Analysis

- ☐ Show All Branches
- ☐ Color By Library
- ☐ Symbol Length Column

Granularity: Symbol
Stats Display: % of Scope
Weight By: Count
Time Unit: Auto
Process ID: PID

▼Callstack Data Mining

- ☑ Apply Callstack Data Mining
- ☐ Charge System Libraries to Callers
- ☑ Charge Code without Debug Info to Callers
- ☐ Hide Weight < 0
- ☐ Flatten Recursion
- ☐ Flatten System Libraries
- ☐ Remove Supervisor Callstacks

Operation	Name

Restore | Unfocus

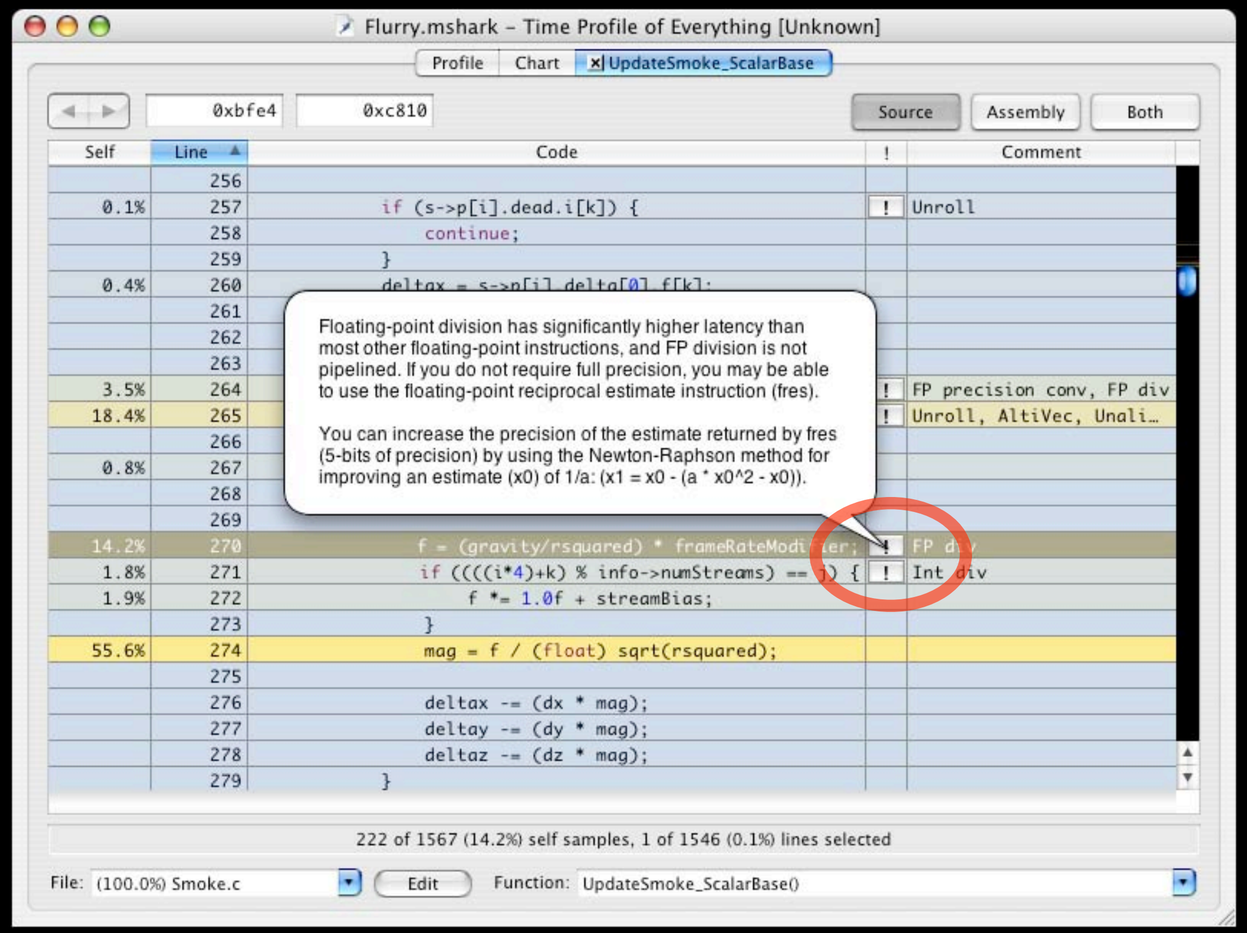
Tuning Your Code

https://developer.apple.com/wwdc2005/

Shark Code Analysis

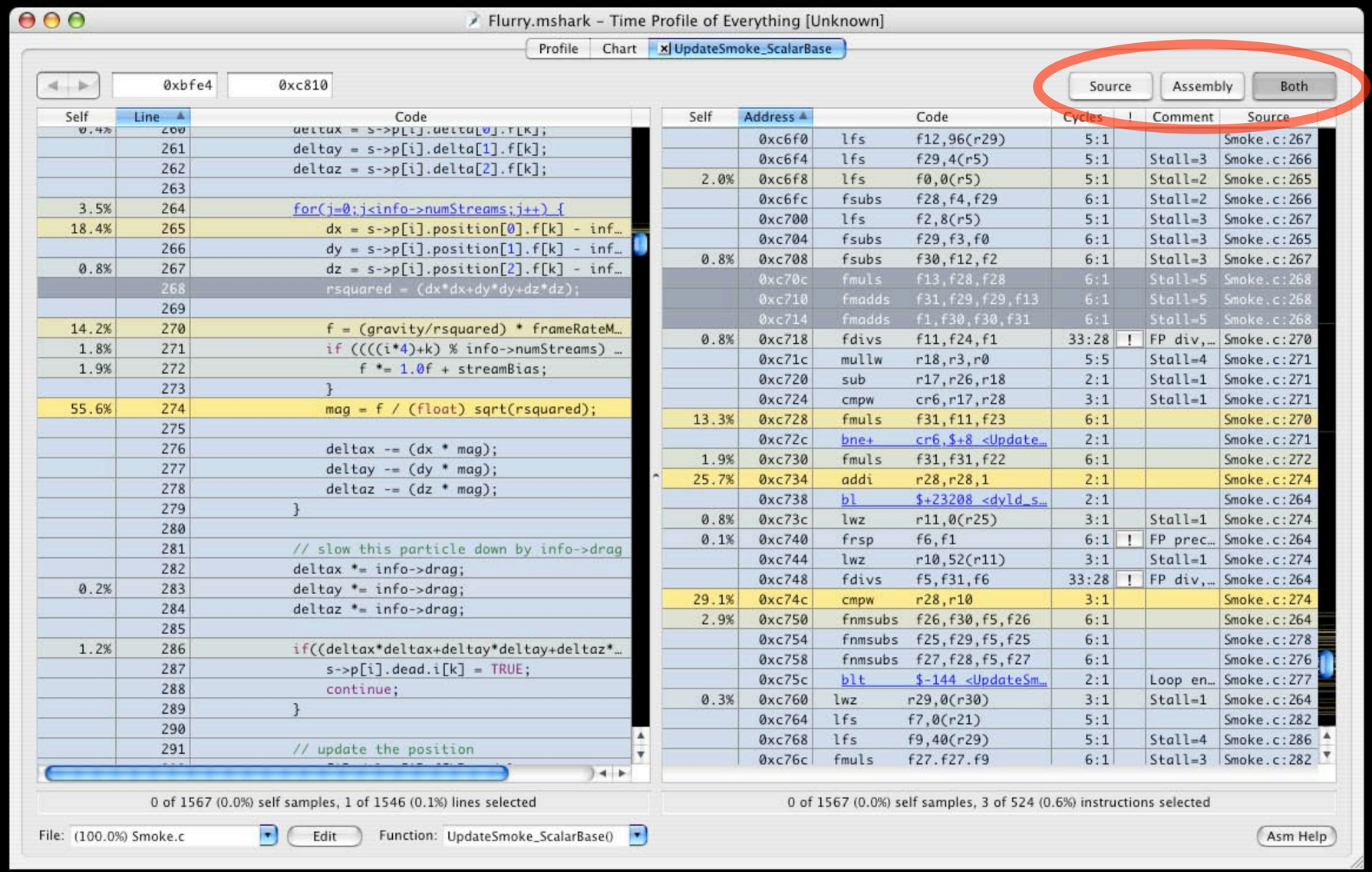
The voice of reason

- Annotates code with optimization tips (“!”)



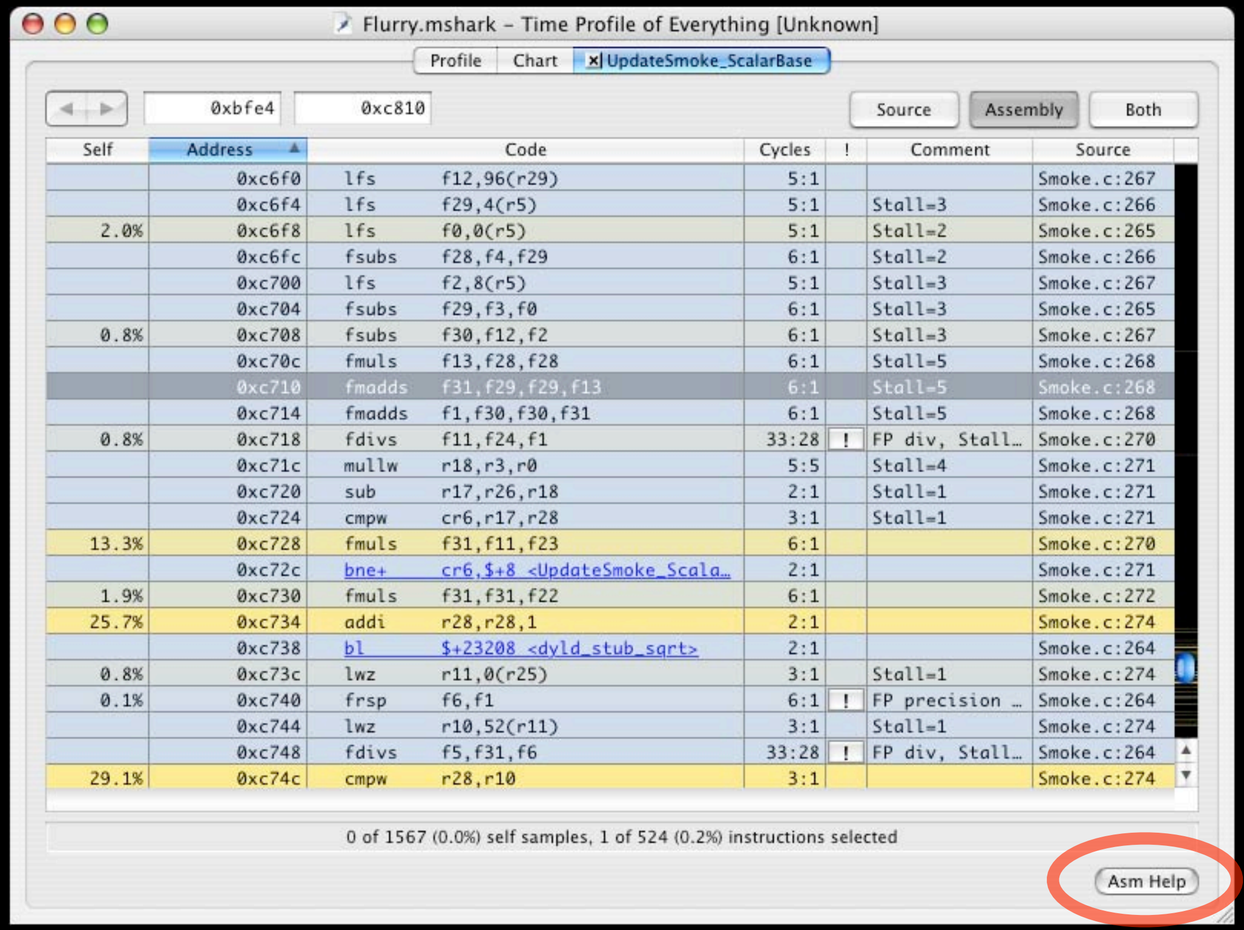
Code Browser

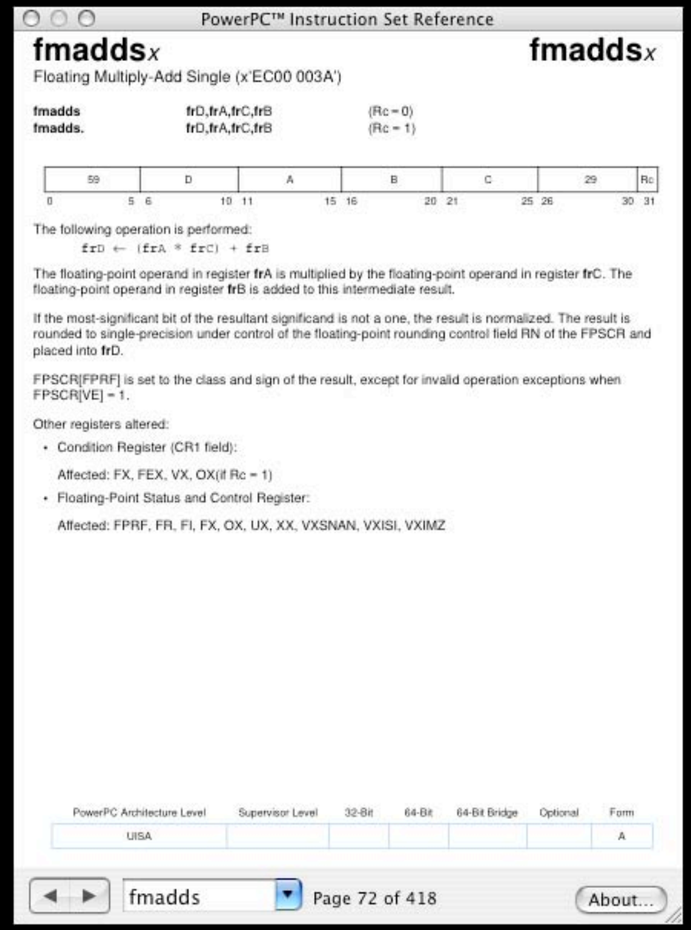
Side-by-side source and assembly!



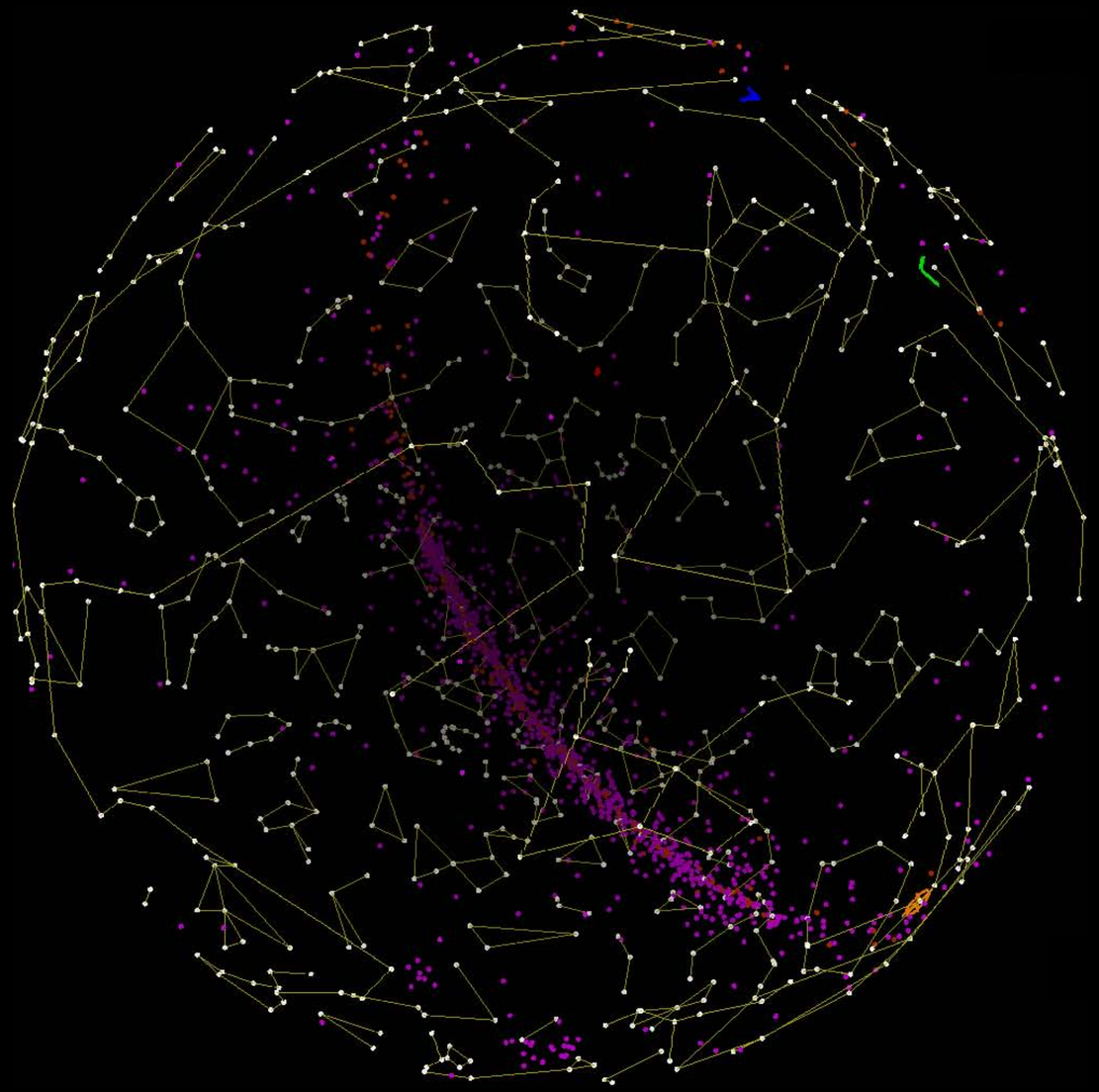
Assembly Help

Down to the metal





Demo



Detect Gravitational Waves Predicted by Albert Einstein
Analyzes LIGO Data on Your Computer
Authors: Bruce Allen, Gaurav Khanna, et. al.

For more information: http://einsteinathome.com

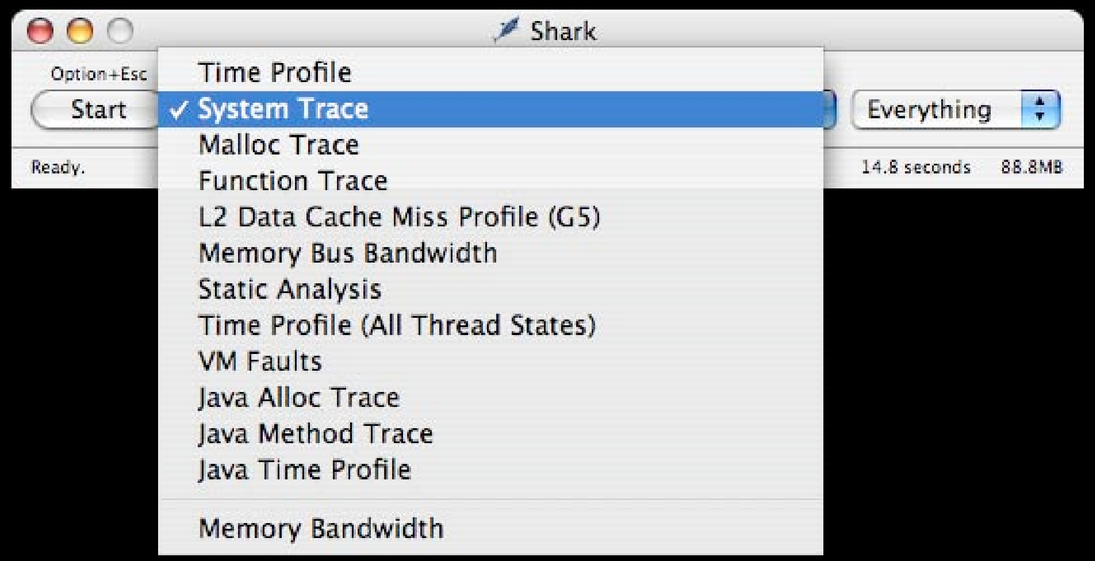
A New Way to Shark

https://developer.apple.com/wwdc2005/

System Trace

System Trace

Tame the Tiger

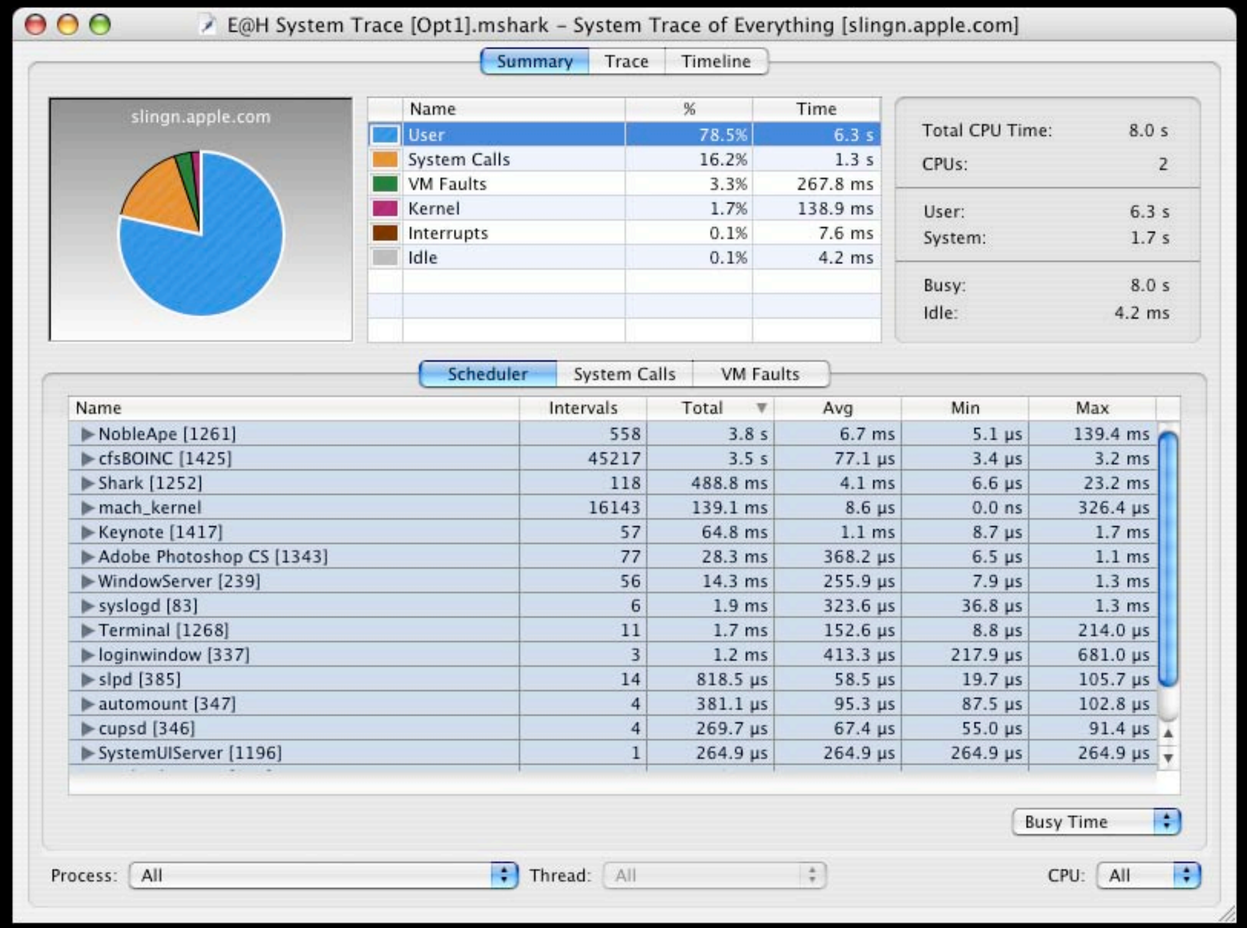


- Goals
  - Understand how your code interacts with Mac OS X
  - Examine multi-threaded behavior
- Methodology
  - Exact trace of boundary crossings
  - No instrumentation required

System Trace

Three ways to see the world

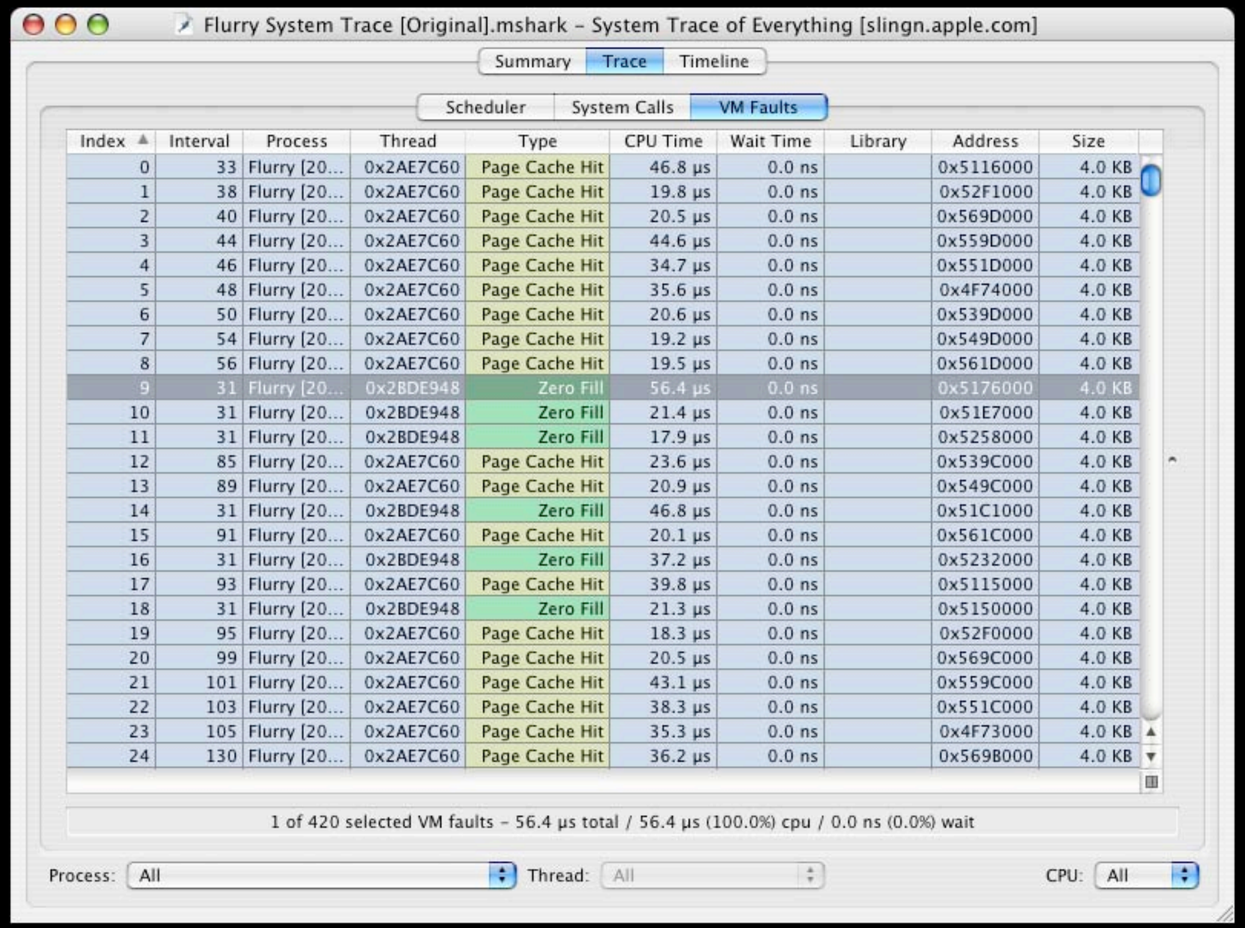
- Summary View



System Trace

Three ways to see the world

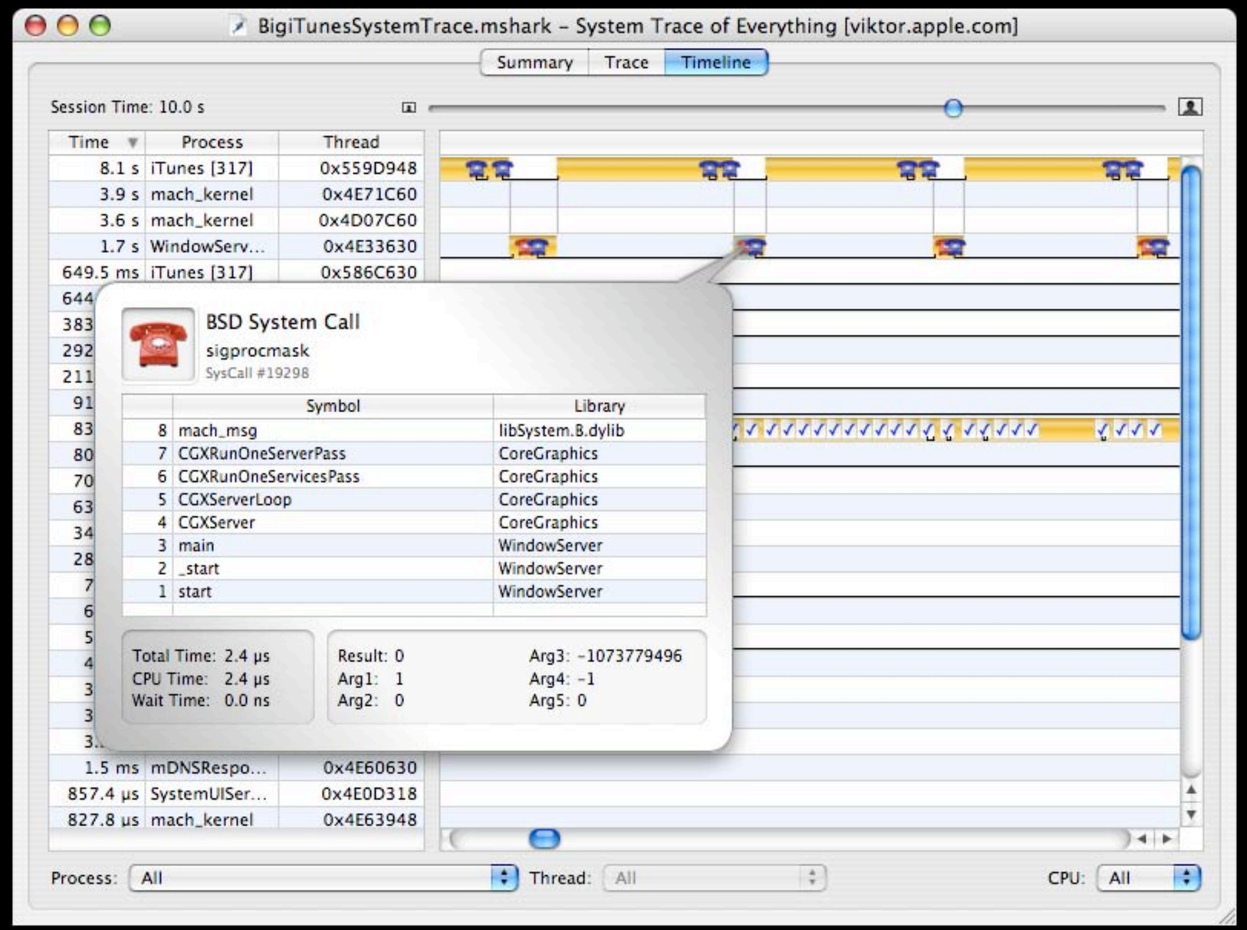
- Summary View
- Trace View



System Trace

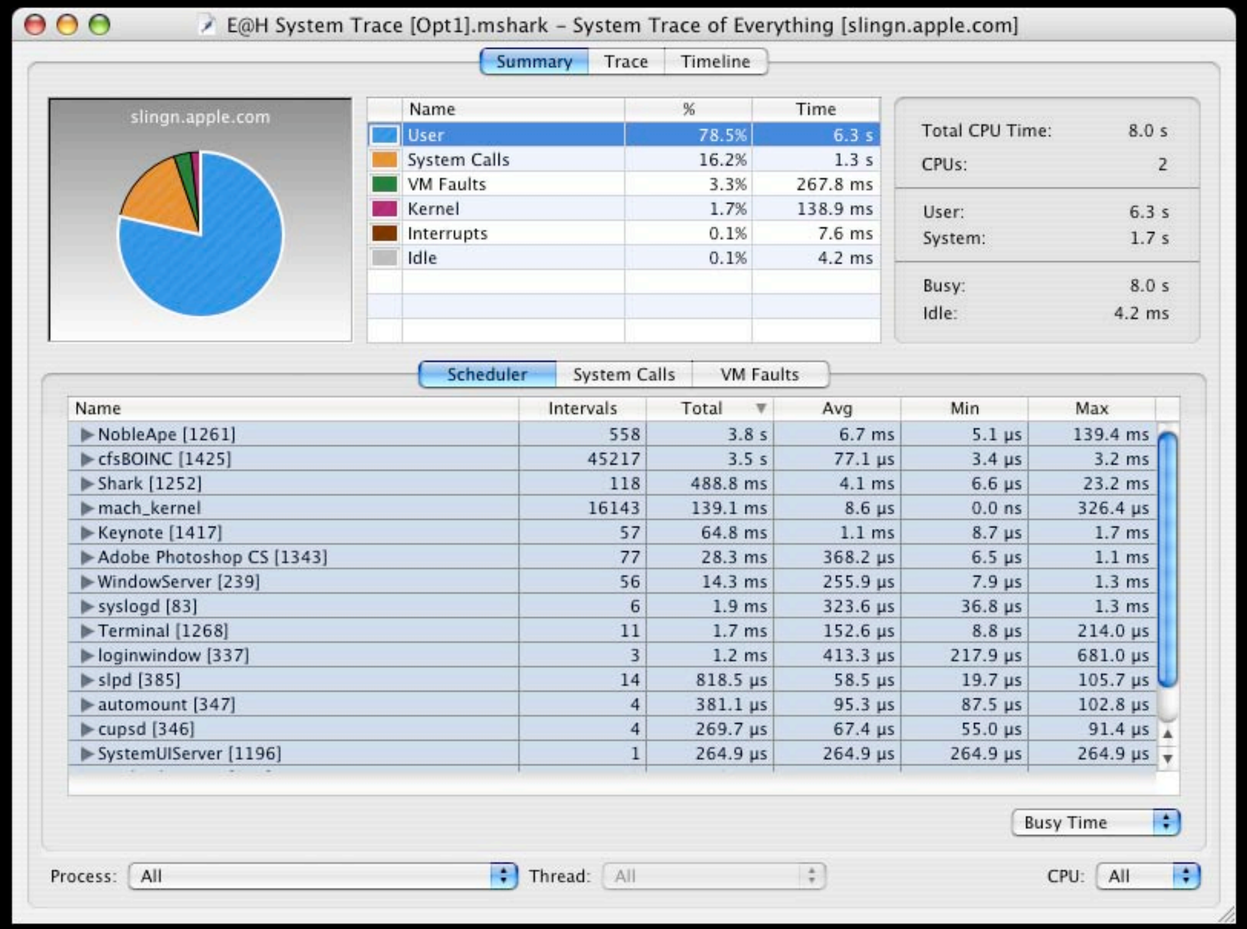
Three ways to see the world

- Summary View
- Trace View
- Timeline View



Summary View

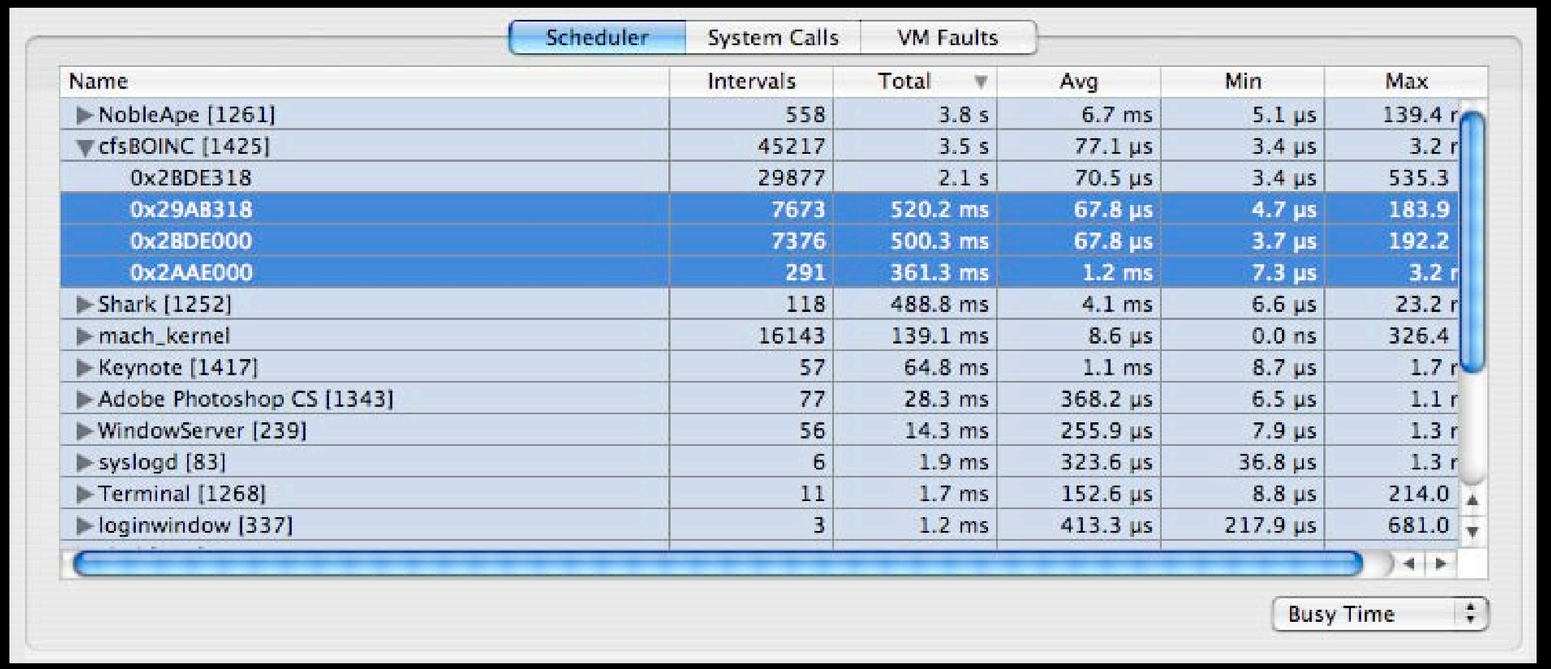
The biggest piece of the pie



Summary View

Scheduler

- Overview of thread scheduling
- View by busy time, user time, system time, or priority

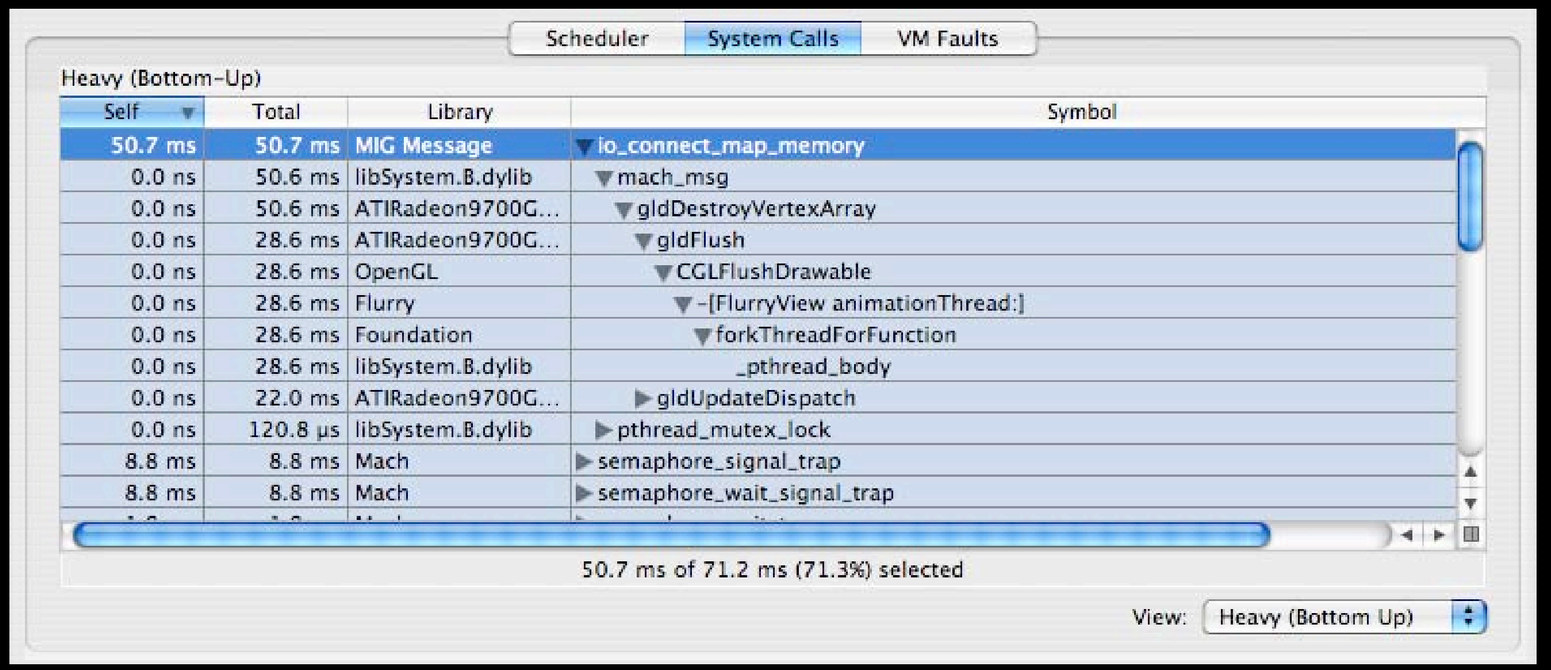


Name	Intervals	Total	Avg	Min	Max
▶ NobleApe [1261]	558	3.8 s	6.7 ms	5.1 µs	139.4 r
▼ cfsBOINC [1425]	45217	3.5 s	77.1 µs	3.4 µs	3.2 r
0x2BDE318	29877	2.1 s	70.5 µs	3.4 µs	535.3
0x29AB318	7673	520.2 ms	67.8 µs	4.7 µs	183.9
0x2BDE000	7376	500.3 ms	67.8 µs	3.7 µs	192.2
0x2AAE000	291	361.3 ms	1.2 ms	7.3 µs	3.2 r
▶ Shark [1252]	118	488.8 ms	4.1 ms	6.6 µs	23.2 r
▶ mach_kernel	16143	139.1 ms	8.6 µs	0.0 ns	326.4
▶ Keynote [1417]	57	64.8 ms	1.1 ms	8.7 µs	1.7 r
▶ Adobe Photoshop CS [1343]	77	28.3 ms	368.2 µs	6.5 µs	1.1 r
▶ WindowServer [239]	56	14.3 ms	255.9 µs	7.9 µs	1.3 r
▶ syslogd [83]	6	1.9 ms	323.6 µs	36.8 µs	1.3 r
▶ Terminal [1268]	11	1.7 ms	152.6 µs	8.8 µs	214.0
▶ loginwindow [337]	3	1.2 ms	413.3 µs	217.9 µs	681.0

Summary View

System Calls

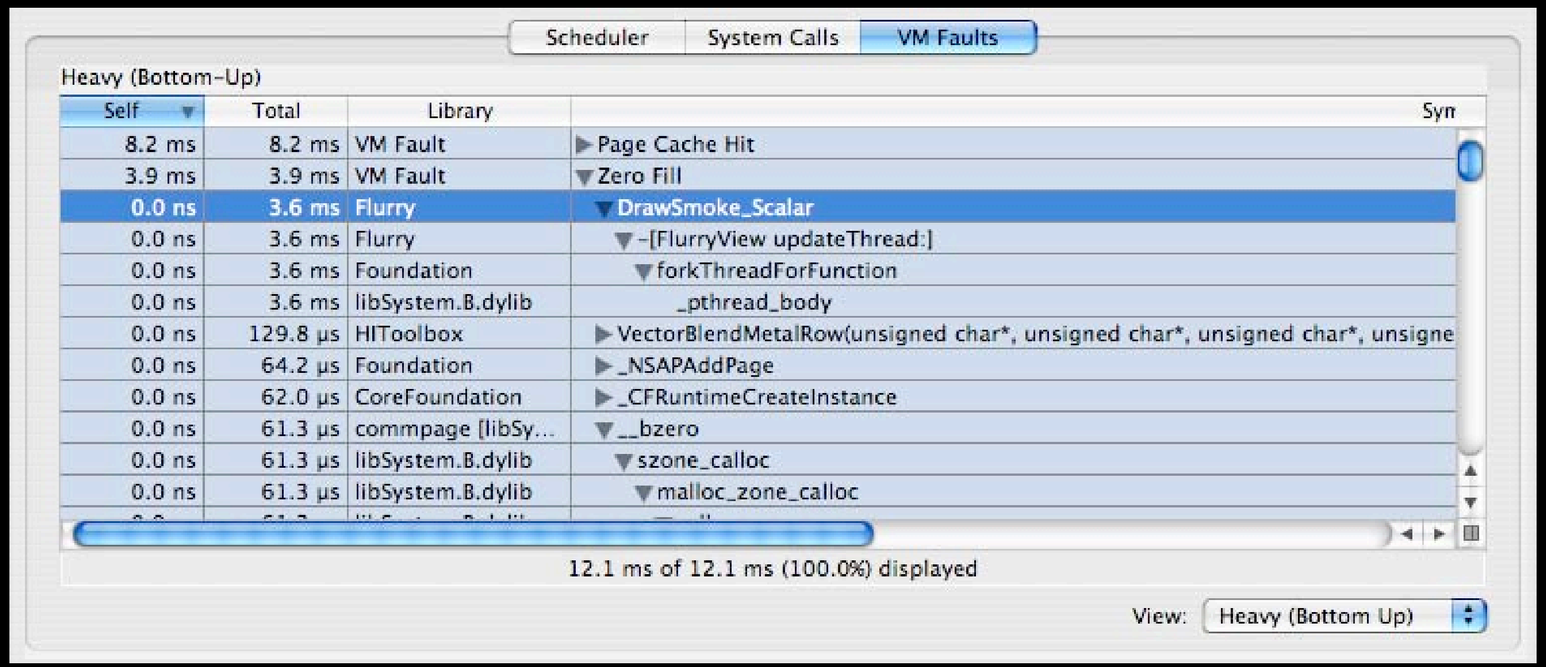
- Profile of system calls
- Attribute system call time to user code



Summary View

VM Faults Calls

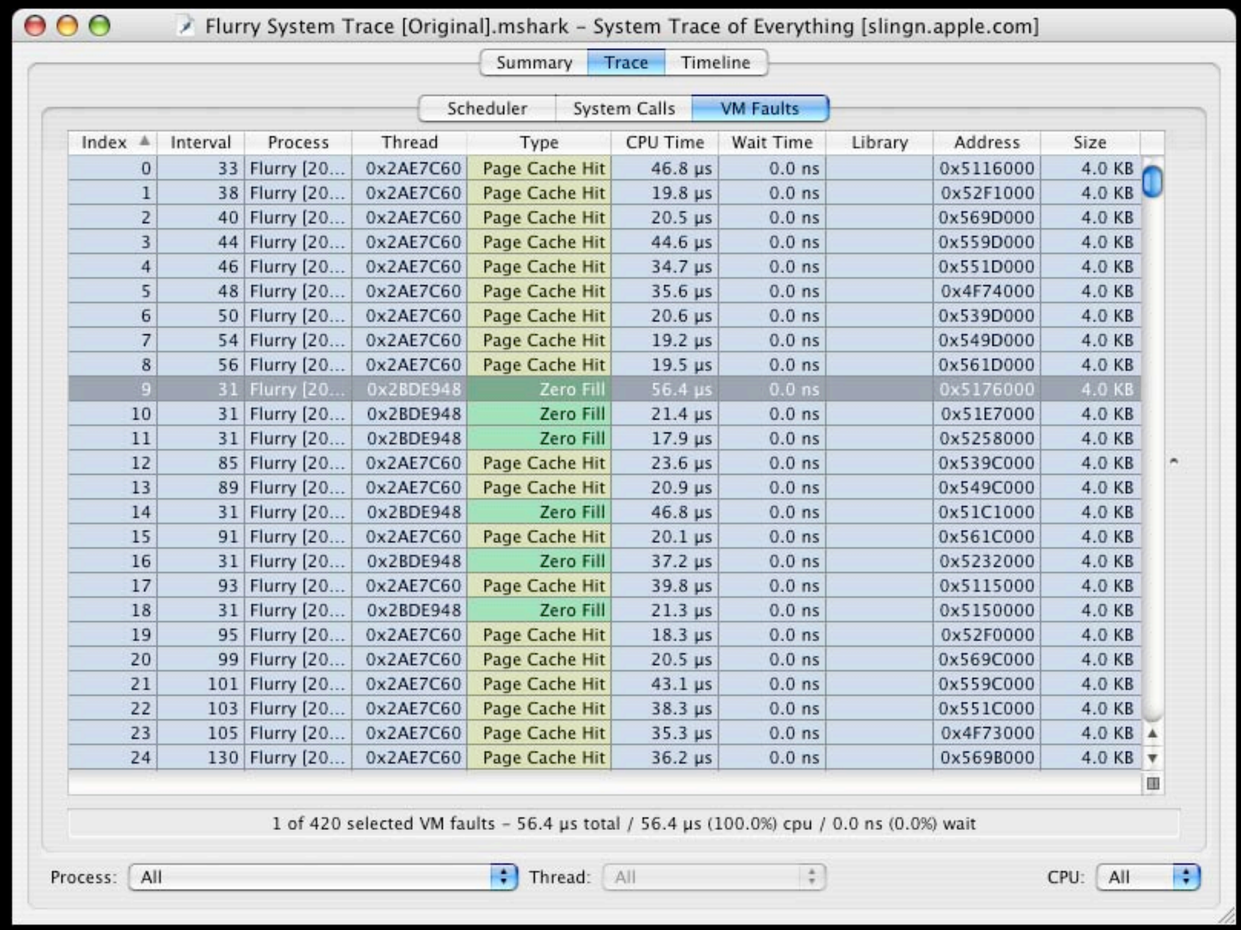
- Profile of VM faults
- Tie virtual memory faults to originating user code



Trace View

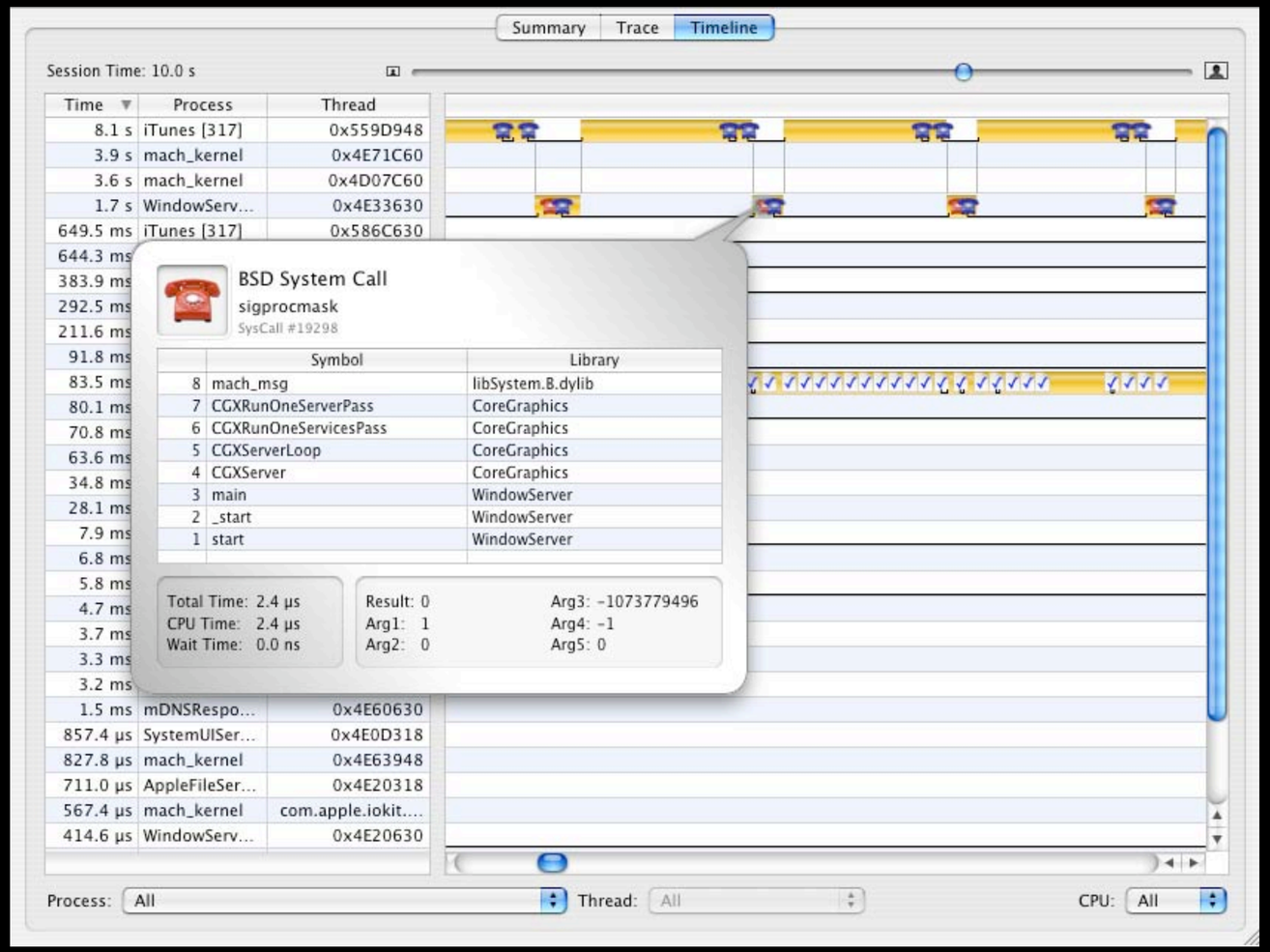
The kitchen sink

- Complete list of traced events
- Inspect callstacks, arguments
- Double click on any event to go to that event in the Timeline View



Timeline View

Nowhere to hide

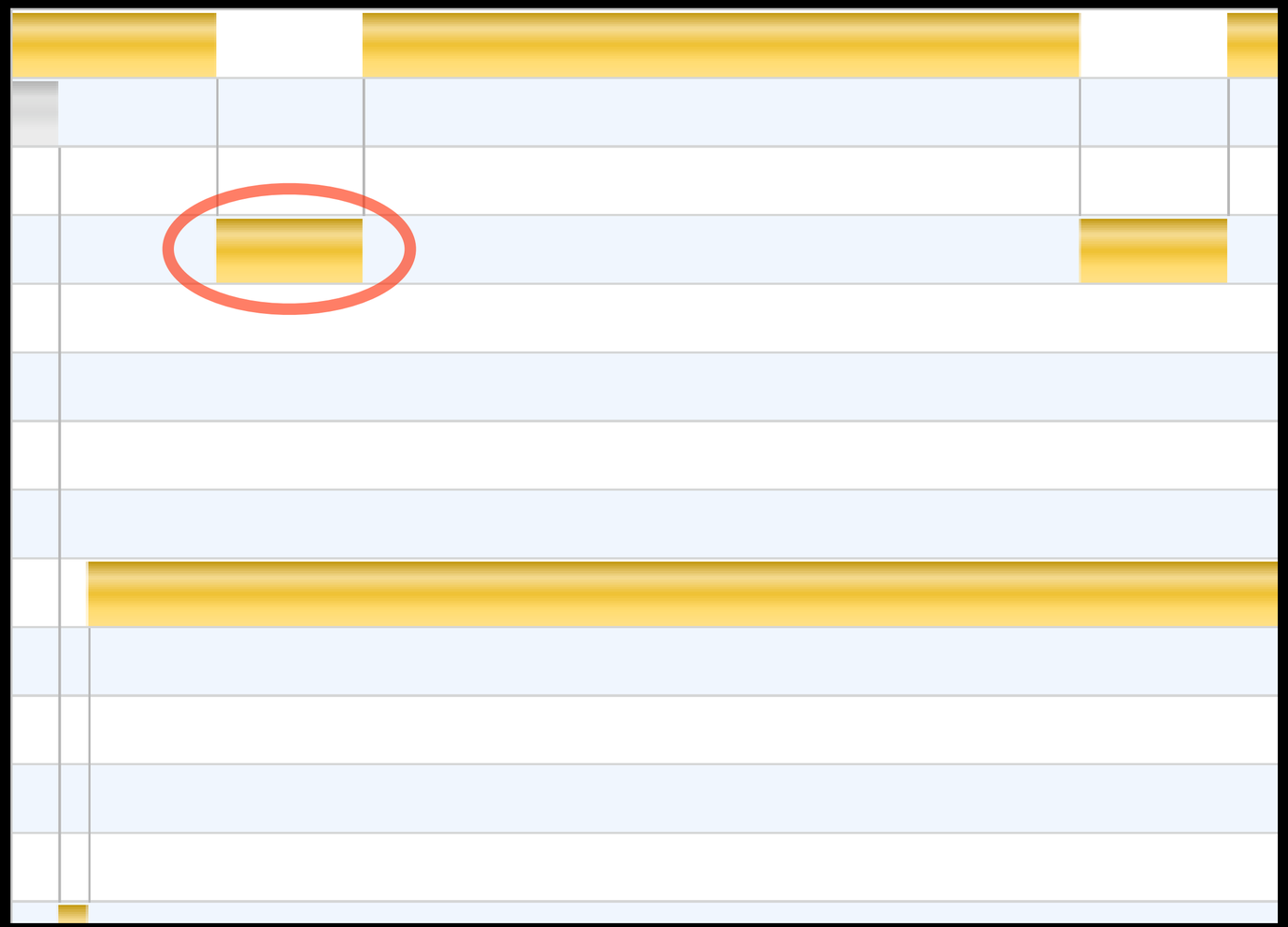


- Navigate
  - Interactive zoom, scroll
  - Inspect events
- Filter
  - System/process/thread
  - CPU
  - Event type
- Color threads
  - CPU
  - Priority
  - Switch reason

Timeline View

Decoder ring

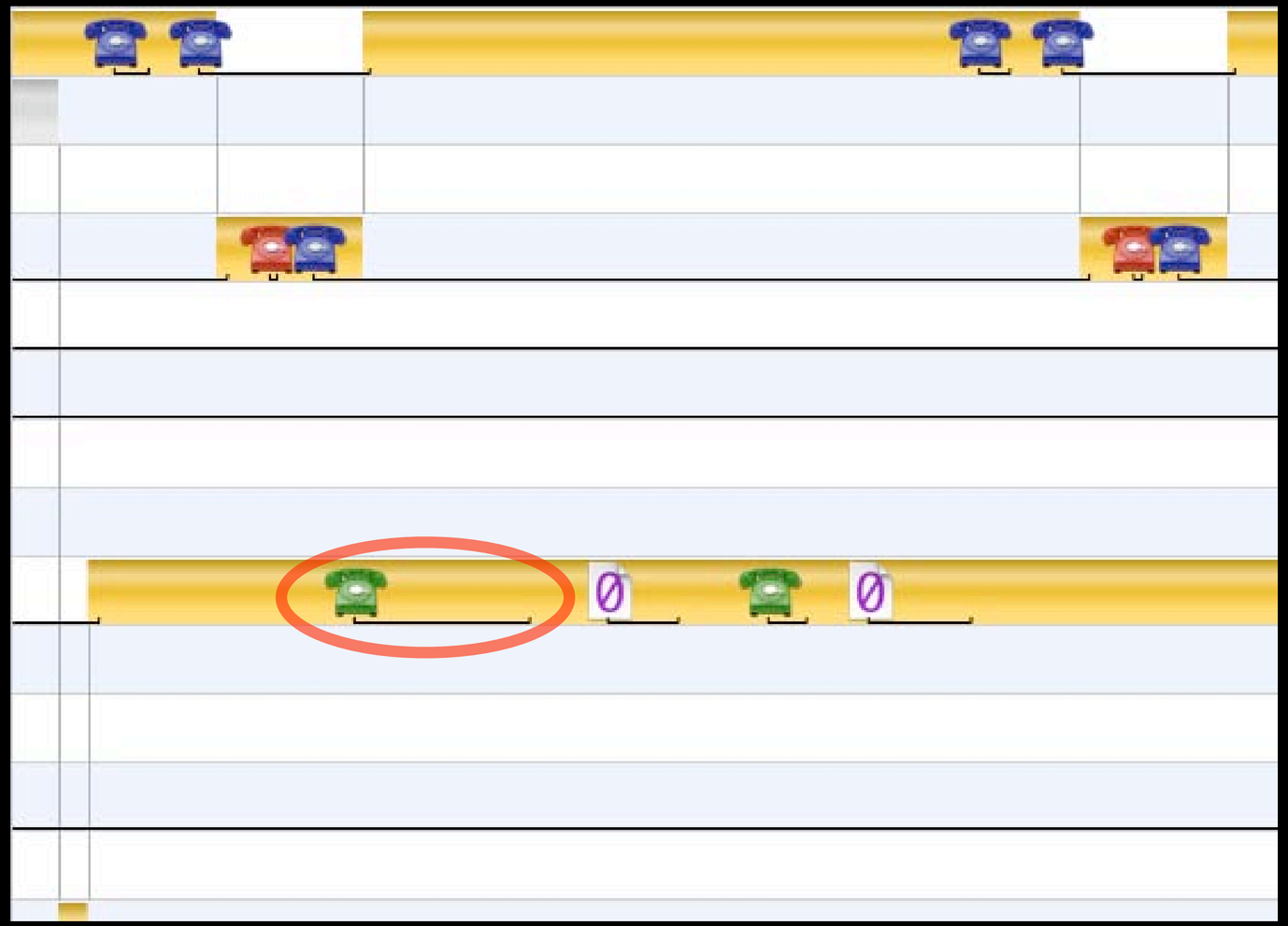
- Thread run intervals



Timeline View

Decoder ring

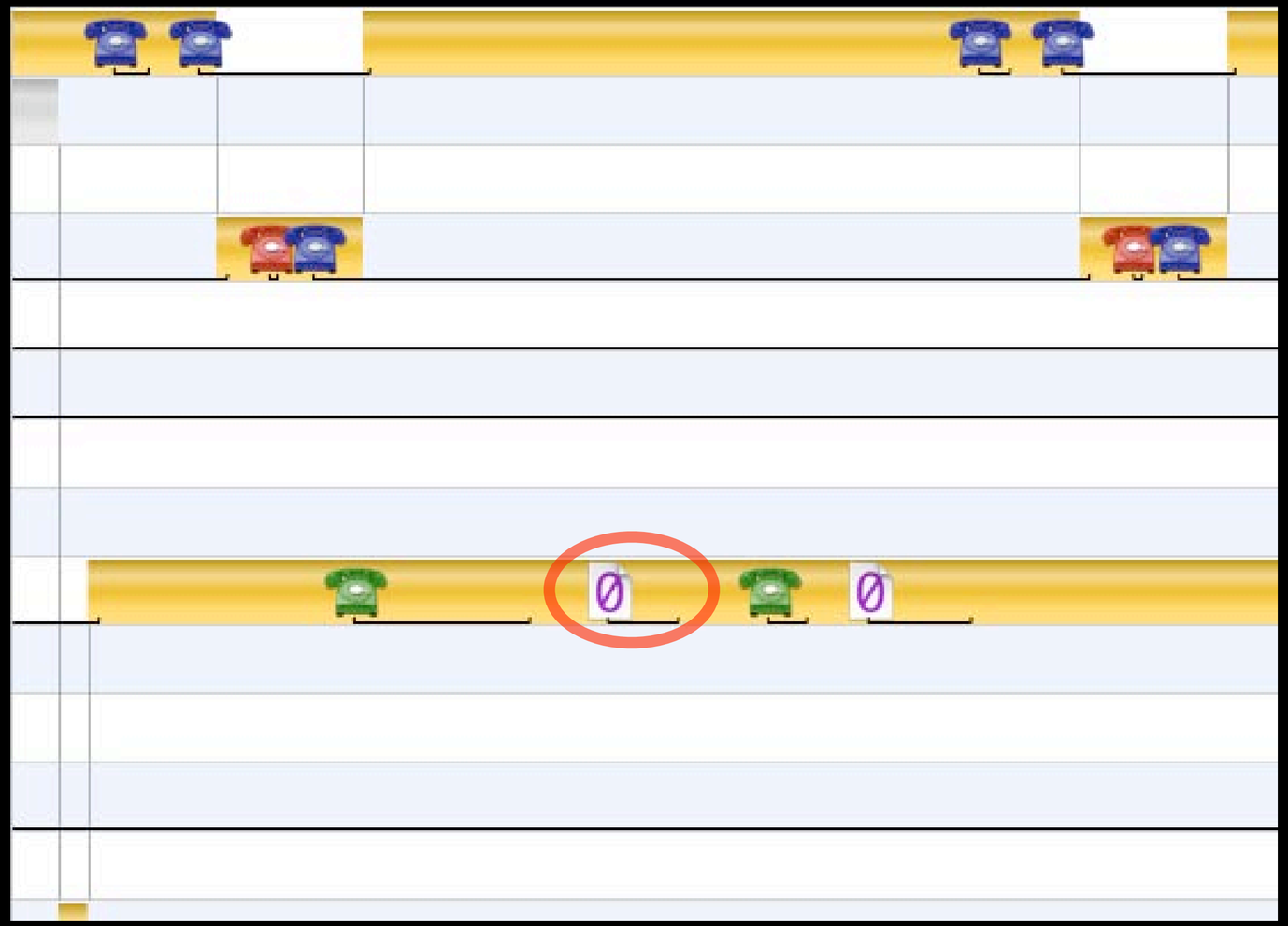
- Thread run intervals
- System calls



Timeline View

Decoder ring

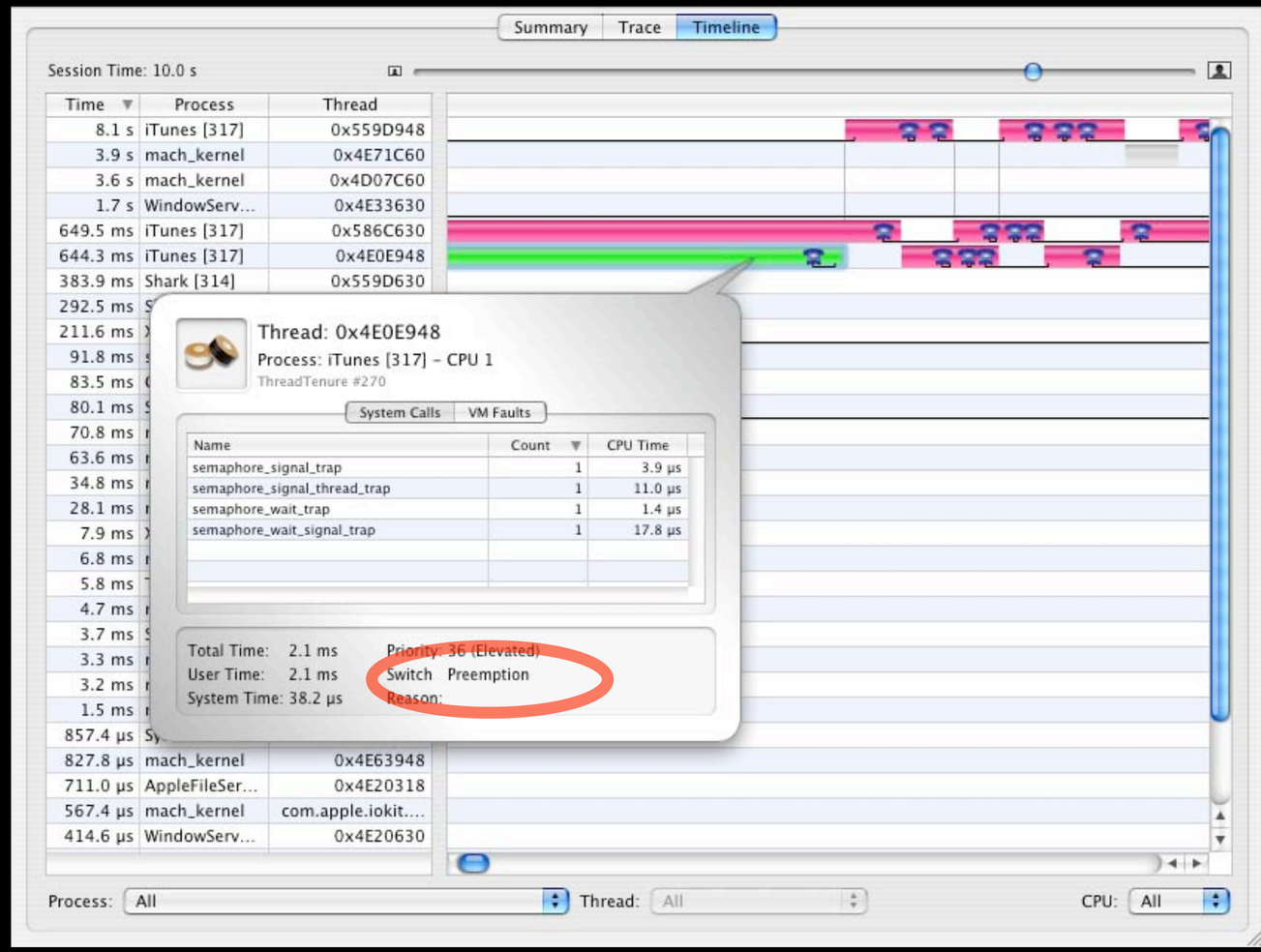
- Thread run intervals
- System calls
- VM faults



Timeline View

Thread run intervals

- Reasons that threads switch
  - Maximum time slice (quantum expired)
  - Waiting on a resource (blocked)
  - Higher priority thread made runnable (preempted)



Timeline View

System calls

- Work done by the kernel on behalf of the calling process
- Allow user space programs to indirectly call kernel APIs

BSD—syscall, ioctl, sysctl APIs

Mach—basic services and abstractions (ports, locks, etc.)

MIG Message—Mach interface generator routines

Timeline View

Virtual memory

- Divides address space into pages
- Pages are brought into physical memory on demand (fault)
- Types of faults

Page In—page pulled from disk (not resident in memory)

Page Out—page pushed out to disk

Zero Fill—page marked zero fill on demand was touched

Copy on Write (COW)—shared page was modified

Page Cache Hit—page resident, but not mapped

Demo

https://developer.apple.com/wwdc2005/

Einstein@Home Performance Summary

	Time (seconds)	Relative Performance
Original	18.6	100%
Opt 1 - Threaded	5508.0	0.3%

Measured on Dual 2.7 GHz G5 PowerMac

Einstein@Home Performance Summary

	Time (seconds)	Relative Performance
Original	18.6	100%
Opt 1 - Threaded	5508.0	0.3%
Opt 2 - Threaded (Large Chunk)	10.6	176%

Measured on Dual 2.7 GHz G5 PowerMac

Einstein@Home Performance Summary

	Time (seconds)	Relative Performance
Original	18.6	100%
Opt 1 - Threaded	5508.0	0.3%
Opt 2 - Threaded (Large Chunk)	10.6	176%
Opt 3 - Unroll, No Division	8.8	211%

Measured on Dual 2.7 GHz G5 PowerMac

Einstein@Home Performance Summary

	Time (seconds)	Relative Performance
Original	18.6	100%
Opt 1 - Threaded	5508.0	0.3%
Opt 2 - Threaded (Large Chunk)	10.6	176%
Opt 3 - Unroll, No Division	8.8	211%
Opt 4 - Single Precision	594.4	3.1%

Measured on Dual 2.7 GHz G5 PowerMac

Einstein@Home Performance Summary

	Time (seconds)	Relative Performance
Original	18.6	100%
Opt 1 - Threaded	5508.0	0.3%
Opt 2 - Threaded (Large Chunk)	10.6	176%
Opt 3 - Unroll, No Division	8.8	211%
Opt 4 - Single Precision	594.4	3.1%
Opt 5 - Smarter Malloc	8.9	209%

Measured on Dual 2.7 GHz G5 PowerMac

Einstein@Home Performance Summary

	Time (seconds)	Relative Performance
Original	18.6	100%
Opt 1 - Threaded	5508.0	0.3%
Opt 2 - Threaded (Large Chunk)	10.6	176%
Opt 3 - Unroll, No Division	8.8	211%
Opt 4 - Single Precision	594.4	3.1%
Opt 5 - Smarter Malloc	8.9	209%
Opt 6 - Vector Hybrid	6.3	295%

Measured on Dual 2.7 GHz G5 PowerMac

Einstein@Home Performance Summary

	Time (seconds)	Relative Performance
Original	18.6	100%
Opt 1 - Threaded	5508.0	0.3%
Opt 2 - Threaded (Large Chunk)	10.6	176%
Opt 3 - Unroll, No Division	8.8	211%
Opt 4 - Single Precision	594.4	3.1%
Opt 5 - Smarter Malloc	8.9	209%
Opt 6 - Vector Hybrid	6.3	295%
Opt 7 - Vector Unroll	5.1	365%

Measured on Dual 2.7 GHz G5 PowerMac

One more thing…

Shark on Intel

Sea change

- Time profiling
- Performance event profiling
- Cross-platform session support
- Code analysis and optimization tips
- Disassemble in the syntax of your choice (Intel or AT&T)
- Integrated instruction reference guide

Demo

https://developer.apple.com/wwdc2005/

Summary

https://developer.apple.com/wwdc2005/

Summary

Swimming with Shark

- Easy for beginners, powerful enough for experts
- Workflows tailored to your needs
  - Time profile
  - System trace
  - Custom configs
- Complete profiling and analysis down to the metal
- CHUD 4.2 Preview available today
  - http://developer.apple.com/tools/download/
- Free!

More Information

https://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Related Sessions

Performance Analysis of Your Memory Code	Session 409 Friday 3:30PM
Development Tools Feedback Forum	Session 410 Friday 5:00PM
Debugging Parallel Applications	Session 643 Friday 5:00PM

Who to Contact

Xavier Legros
Mac OS X Technology Evangelist
xavier@apple.com

Performance Tools Feedback List
perftools-feedback@group.apple.com

Q&A

https://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference